



Karlsruher Institut für Technologie
Fakultät für Informatik



Institut für Angewandte Informatik und
Formale Beschreibungsverfahren

Semiautomatische Anreicherung einer semantischen virtuellen Forschungsumgebung mit externen Daten

Diplomarbeit

von

Jan Novacek

Eingereicht am 15. Juli 2014

bei dem

Institut für Angewandte Informatik und
Formale Beschreibungsverfahren (AIFB)
am Karlsruher Institut für Technologie

Gutachter: Prof. Dr. Rudi Studer
Betreuender Mitarbeiter: M.Sc. Basil Ell

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche gekennzeichnet. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Speyer, den 14. Juli 2014

Jan Novacek

Zusammenfassung

Im Rahmen dieser Arbeit sollte eine Lösung entwickelt werden, die ForscherInnen bei ihrer Arbeit mit einer Virtuellen Forschungsumgebung, unterstützt. Der am Beispiel der Virtuelle Forschungsumgebung entwickelte Lösungsansatz und dessen Implementierung kann in jede ViFU integriert werden, die auf MediaWiki basiert. Durch den modularen Aufbau können Teile der Architektur leicht ersetzt, erweitert oder in anderer Software wiederverwendet werden. Der Entwicklungsprozess bezog ForscherInnen als Benutzer der ViFU mit ein und wurde so ausgelegt, dass eine langfristige, gemeinschaftliche Entwicklung im Geiste der MediaWiki Gemeinschaft möglich wäre. Die Benutzbarkeit der Softwarelösung wurde als wichtiger Gestaltungsgrundsatz in der Anforderungsanalyse und der Evaluation identifiziert und konnte mit positiven Ergebnissen realisiert werden.

Danksagungen

Ich möchte meiner Familie für die Unterstützung bei der Anfertigung meiner Diplomarbeit und das viele Interesse danken. Besonders Danken möchte ich meiner Mutter, die mir in vielen Stunden beim Korrekturlesen zur Seite stand und mir mit den richtigen Worten stets moralischen Rückenwind bot. Auch meinem Vater möchte ich für sein Interesse an meiner Arbeit danken, sowie meinem Onkel für die wertvollen Anregungen.

Nicht zuletzt möchte ich mich auch bei meinem Betreuer für die hilfreiche und fürsorgliche Begleitung meiner Arbeit bedanken, der mir mit vielen Ratschlägen und Diskussionen über den gesamten Zeitraum zur Seite stand und mir bei vielen Herausforderungen weiterhelfen konnte.

An dieser Stelle möchte ich außerdem den freundlichen und hilfsbereiten Mitarbeitern des Instituts für Angewandte Informatik der Universität Leipzig für ihre Unterstützung bei der Integration von LIMES in die MediaWiki Erweiterung sehr danken.

Danken möchte ich auch allen Beteiligten am Projekt Semantic CorA, mit denen ein gemeinsamer, produktiver Austausch im Rahmen der Anforderungsanalyse stattfand und die auch als Probanden im Rahmen der Evaluation dieser Arbeit wertvolles Feedback beigetragen haben.

Inhaltsverzeichnis

1	Einführung	1
1.1	Motivation	1
1.1.1	Beispielszenario	2
1.2	Ziel	2
1.3	Beitrag	3
1.4	Gliederung	3
2	Grundlagen	4
2.1	Semantic Web	4
2.1.1	Linked Data	5
2.1.2	Semantic MediaWiki (SMW)	8
2.2	Virtuelle Forschungsumgebungen (ViFU)	12
2.2.1	Beispiele für ViFUen	12
2.3	Entity Matching (EM)	16
2.3.1	Hintergrund	16
2.3.2	Problem Definition	18
2.3.3	Anwendungen und Herausforderungen	19
2.3.4	Ansätze und Aspekte	21
2.4	Datenintegration	26
2.4.1	Definition	26
2.4.2	Datenintegration und Linked Data	26
2.4.3	Datenintegration und ViFUen	26
3	Architektur	27
3.1	Anforderungen	27
3.1.1	Anforderungsanalyse	27
3.1.2	Funktionale Anforderungen	29
3.1.3	Nichtfunktionale Anforderungen	30
3.1.4	Randbedingungen	31
3.2	Verwandte Arbeit	31
3.3	Lösungsansatz	32
3.3.1	Alternativen	34
3.3.2	Architektur Auswahl	35

3.3.3	Komponenten und Aufbau der Architektur	36
4	Implementierung	42
4.1	Überblick	42
4.1.1	Vorgehensmodell	43
4.1.2	Eingesetzte Technologien	44
4.1.3	Herausforderungen der Implementierung	44
4.2	Implementierung des Back-Ends	44
4.2.1	Entity Matching Frameworks	45
4.2.2	Datenquellen	47
4.2.3	Datenintegration	49
4.2.4	Serverseitige API	51
4.3	Implementierung des Front-Ends	52
4.3.1	Clientseitige API	52
4.3.2	Grafische Benutzerschnittstelle (GUI)	52
4.4	Implementierung spezieller Werkzeuge	54
4.5	Schwierigkeiten	55
5	Evaluation	57
5.1	Ziele der Evaluation	57
5.2	Methodik	57
5.3	Ergebnisse	59
5.3.1	Erfüllung der Zielsetzung	59
5.3.2	Abdeckung von Anforderungen	59
5.3.3	Gebrauchstauglichkeit	60
5.3.4	ISONORM 9241/110	62
5.4	Offene Probleme	64
5.4.1	Funktionalität der Datenintegration	64
5.4.2	Weitere EM Frameworks	65
5.4.3	Erweiterung der Konfigurationsmöglichkeiten	65
5.4.4	Herkunft von Daten	65
6	Schluss	67
6.1	Zusammenfassung	67
6.2	Ausblick	67
6.2.1	Kandidaten für Referenzverknüpfungen	67
6.2.2	EM Crowdsourcing	68
A	Abkürzungsverzeichnis	69
B	Quelltexte	71
C	Grafische Benutzerschnittstelle	72
D	Anforderungsanalyse	75

E	Liste betrachteter Entity Matching Frameworks	77
	Literatur	79
	Index	89

Kapitel 1

Einführung

Wikis haben sich als Systeme zum Sammeln und der Verwaltung von Wissen in vielen Bereichen etabliert. Diese Entwicklung hat auch in der Wissenschaft Einzug gehalten, wo Wikis auch als *Virtuelle Forschungsumgebungen* verwendet werden. Diese Arbeit behandelt die Integration von *Entity Matching* Verfahren in Virtuelle Forschungsumgebungen zur Unterstützung von Forschern beim Anreichern von Entitäten mit Daten aus externen Datensätzen.

1.1 Motivation

Wie zuvor erwähnt werden Wikis auch als *Virtuelle Forschungsumgebung* (ViFU) in der Wissenschaft verwendet, was im Abschnitt 2.2 näher beschrieben wird. Bei der Verwendung der ViFU kann sich unter anderem die Herausforderung stellen, dass vorhandene Entitäten mit weiteren Daten angereichert werden sollen. Je nach Anzahl und Art der Entitäten und der anzureichernden Daten kann dies einen hohen Arbeitsaufwand verursachen. In Abhängigkeit von den Bedürfnissen der Benutzer der ViFU und in Hinsicht auf die Methodik zur Bearbeitung der Forschungsfrage, kann das Problem durch die Identifikation korrespondierender Entitäten in externen Datenbanken und dem Hinterlegen von Verweisen in der ViFU auf diese Entitäten, gelöst werden.

Werden Daten dieser Entitäten aber beispielsweise von einem Werkzeug innerhalb der ViFU zu dessen Ausführung benötigt, so müssen diese Daten zusätzlich in die ViFU integriert werden.

Zur Verdeutlichung der Problemstellung soll nachfolgend noch ein Beispiel für ein Szenario angeführt werden.

1.1.1 Beispielszenario

Eva hat im Rahmen ihrer wissenschaftlichen Tätigkeit in der ViFU Semantic CorA Daten über 4041 Personen gesammelt, die für ihr Projekt relevant sind. Diese Daten umfassen den Vor- und Nachnamen, die ausgeübte Tätigkeit und weitere personenbezogene Daten. Allerdings verfügen nicht alle Personen über die gleiche Anzahl eingetragener Daten.

Um ein neues Problem bearbeiten zu können, das im Rahmen ihres Projektes aufgetreten ist stellt Eva fest, dass sie neben den Daten die sie bereits eingetragen hat, noch den Geburtsort und das Geburtsdatum der Personen benötigt, die Gegenstand ihrer Forschung sind. Nun steht Eva vor dem Problem, diese Daten bei allen 4041 Personen einzeln nachtragen zu müssen.

Sie verwendet also die Erweiterung der ViFU, die im Rahmen dieser Arbeit entstanden ist, um fehlende Daten aus den Einträgen zu den entsprechenden Personen aus der Datenbank der Deutschen Nationalbibliothek zu beziehen. Um dies zu erreichen, wählt Eva zunächst 1056 Personen aus, bei denen die fehlenden Daten ergänzt werden sollen. Anschließend gibt Eva an, dass diese Daten aus der Datenbank der Deutschen Nationalbibliothek zu beziehen sind. Anschließend bestätigt Eva die von ihr gewählten Einstellungen. Sobald die Einstellungen bestätigt wurden, startet der Prozess zur Identifikation der Personen in dem Datensatz der Deutschen Nationalbibliothek und Eva kann ihre Arbeit bis die Ergebnisse vollständig zur Verfügung stehen an anderer Stelle fortsetzen.

Ist der Vorgang vollständig abgeschlossen, so kann Eva die Erweiterung veranlassen, Verknüpfungen zu den Entitäten der Deutschen Nationalbibliothek den Entitäten der Virtuellen Forschungsumgebung hinzuzufügen.

Darüber hinaus kann Eva zusätzlich auswählen, dass Geburtsort, sowie Geburtsdatum den zuvor ausgewählten Personen hinzugefügt werden sollen.

1.2 Ziel

Aus der Motivation und dem an der Praxis orientierten Beispielszenario heraus wurde dieser Arbeit zu Beginn das Ziel gesetzt, eine Lösung für eine konkrete ViFU zu erarbeiten. Die wesentlichen Aspekte, die dabei bearbeitet werden sollten sind:

- Semi-automatische Identifikation von korrespondierenden Entitäten einer ViFU und der eines externen Datensatzes.
- Semi-automatische Integration der Daten von identifizierten Entitäten eines externen Datensatzes in eine ViFU.
- Benutzerfreundliche Gestaltung der Realisierung, zur Unterstützung der Forscher bei der Arbeit in Hinsicht auf die beiden vorherigen Aspekte.

1.3 Beitrag

Die Ergebnisse dieser Arbeit werden in Kapitel 5 genauer beschrieben. Zusammenfassend sind die Beiträge dieser Arbeit folgende:

- Unterstützung des *Semantic Web* Konzepts, nach dem Prinzip der Anreicherung von Daten mit Metadaten.
- Es wird eine Übersicht über Entity Matching Ansätze geboten, sowie eine Liste von Entity Matching Frameworks, die bei den Recherchen identifiziert werden konnten.
- Möglichkeiten zur Integration von Entity Matching Ansätzen in ViFUen, die auf MediaWiki bzw. Semantic MediaWiki basieren, werden analysiert.
- Realisierung einer Lösung zur Erreichung der Zielsetzung durch die Integration von Entity Matching in eine ViFU in Form einer Erweiterung für MediaWiki bzw. Semantic MediaWiki.

1.4 Gliederung

Dieses Dokument ist in 6 Kapitel unterteilt. Der Einführung, in welcher die Arbeit motiviert und die Zielsetzung formuliert wurde, folgt ein Kapitel in dem Grundlagen behandelt werden, die für diese Arbeit relevant sind. Das dritte Kapitel stellt dann den Lösungsansatz zur Erreichung der Zielsetzung vor, was anschließend im vierten Kapitel mit der Beschreibung der Implementierung anhand des praktischen Teils vertieft wird. Abschließend werden dann die Ergebnisse dieser Arbeit evaluiert und Schlußfolgerungen gezogen.

Kapitel 2

Grundlagen

Dieses Kapitel behandelt Grundlagen, die mit dem Thema dieser Arbeit im Zusammenhang stehen. Zunächst wird in das Thema *Semantic Web* eingeführt. Anschließend werden *MediaWiki* und dessen Erweiterung *Semantic MediaWiki* beschrieben und abschließend folgen Abschnitte über Virtuelle Forschungsumgebungen, Entity Matching und Datenintegration.

2.1 Semantic Web

Erste Bestrebungen, das Internet zu einem Semantischen Netz aufzuwerten, reichen in das Jahr 1998 zurück [BL98]. Die Repräsentationen von Informationen im Internet wurden so entworfen, dass diese für Menschen, aber nicht zur Verarbeitung durch Maschinen geeignet waren. Beispielsweise würde sich eine Suche auf Wikipedia nach allen Musikern, die vor dem Jahr 2000 geboren wurden und keine Schlagermusiker sind, als umständlich gestalten. Da sich eine solche Liste jener Musiker zwar mit Hilfe der in den Artikeln der Wikipedia Enzyklopedie intrinsischen Daten prinzipiell erstellen ließe, die Verarbeitung dieser Aufgabe durch eine Maschine aber erst durch die explizite Angabe jener Daten ermöglicht oder zumindest stark vereinfacht würde.

Neben der expliziten Angabe von Daten, was im Internet beispielsweise durch Markup-Sprachen wie XML oder HTML realisiert wird, kann die Verarbeitung von Menschen geschriebener Texte durch Maschinen außerdem durch die Definition der Semantik der zu verarbeitenden Daten ermöglicht werden. In Bezug auf das Beispiel zuvor, wäre es bspw. nicht aufreichend nur Datumsangaben in einem Wikipedia Artikel als solche zu markieren, da dann nicht entschieden werden könnte, ob es sich bspw. um ein Geburtsdatum oder Sterbedatum handelt.

Die Definition der Semantik von Geburts- und Sterbedatum ermöglicht nicht nur die Unterscheidung zweier konkreter Datumsangaben, sondern auch diese in Relation zu setzen, wie beispielsweise über Geburts- und Sterbedatum

eine Ordnungsrelation zu definieren und implizite Informationen zu inferieren wie bspw. die dominierenden musikalischen Stilrichtungen verschiedener Epochen.

Die Herausforderung des Semantic Web Ansatzes liegt nun darin, Sprachen zu entwickeln die es ermöglichen, sowohl Daten explizit anzugeben, sowie deren Semantik zu definieren [BLHL⁺01]. Abbildung 2.1 zeigt Illustrationen verschiedener bekannter Architekturmodelle des Semantic Web und stellt auch die Hierarchie bereits existierender Sprachen dar. Außerdem soll durch die vier Modelle verdeutlicht werden, dass die Architektur nach wie vor diskutiert und weiterentwickelt wird.

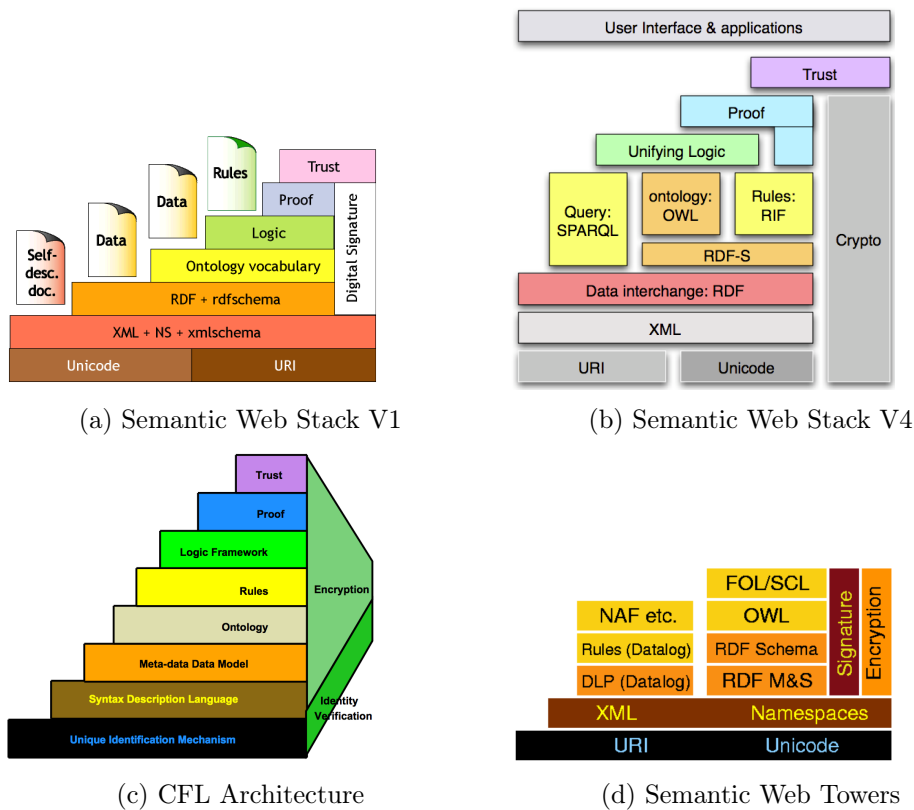


Abbildung 2.1: Übersicht über verschiedene Vorschläge zur Architektur des Semantic Web. (a) und (b) stammen von Tim Berners-Lee [BL00, BL06a], (c) nach Gerber et al. [GBVdM07, GvdMB08] und (d) stammt von Ian Horrocks et al. [HPPSH05].

2.1.1 Linked Data

In diesem Abschnitt wird das *Linked Data* oder auch *Linked Open Data* (LOD) Konzept vorgestellt, das in Zusammenhang mit den Semantic Web Bestrebungen steht und überdies im selben Zusammenhang ein Anwendungsfeld für Entity Matching darstellt, was in Abschnitt 2.3 behandelt wird. Weiterhin ist Linked Data für die Integration von Daten in Virtuelle Forschungsumgebungen von Bedeutung, was im späteren Abschnitt 2.3.1 verdeutlicht wird.

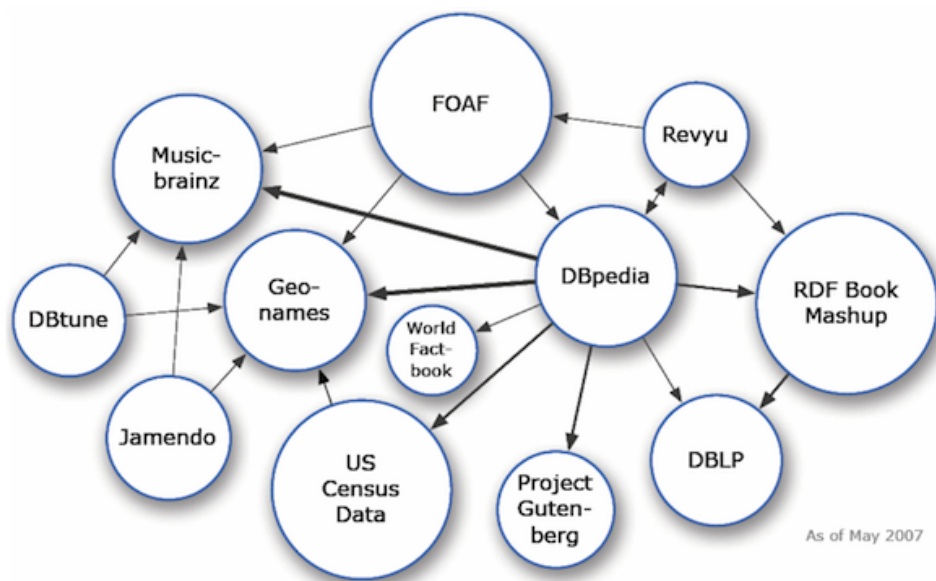
Zusammenfassend ausgedrückt, bedeutet Linked Data einfach das Internet zu verwenden, um Links mit Typenangabe zwischen Daten aus verschiedenen Quellen zu erstellen [BHBL09]. Der Zusammenhang mit den Semantic Web Bestrebungen besteht darin, dass durch die Angabe des Typs eines Links, eine semantische Relation spezifiziert wird. Dies geschieht durch Formulierung von Aussagen unter Verwendung von RDF.

In [BL06b] wurde eine Menge von Regeln zum Veröffentlichen von Daten im Internet skizziert, deren Einhaltung das Verbinden von Daten unterschiedlicher Quellen verbessert:

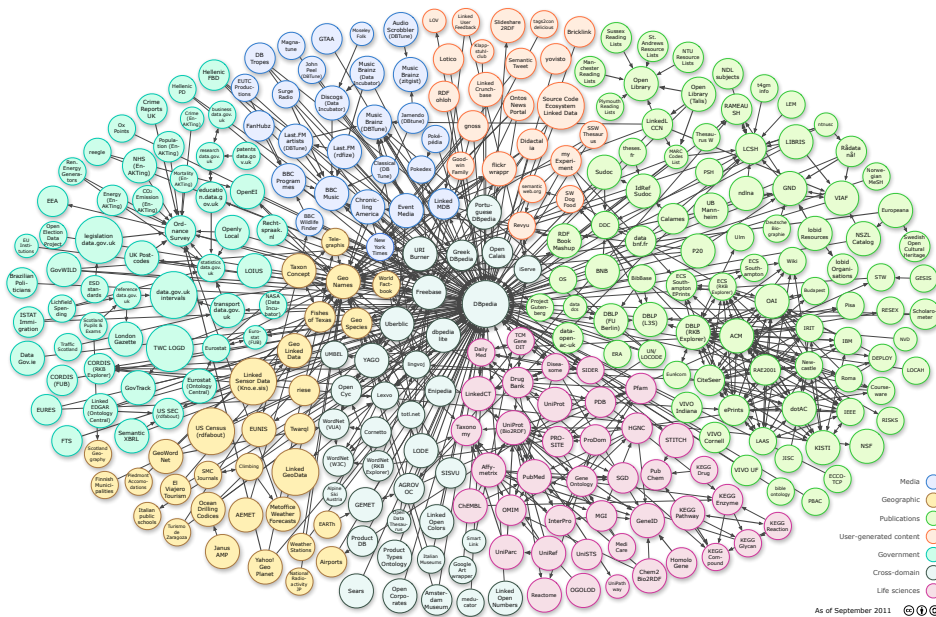
1. URIs als Namen für Objekte verwenden.
2. HTTP URIs verwenden, sodass Benutzer diese Namen nachschlagen können.
3. Wird eine URI nachgeschlagen, so sind nützliche Informationen in standardisierter Form über RDF und SPARQL bereitzustellen.
4. Einbindung von Links zu anderen URIs, sodass weitere Objekte entdeckt werden können.

Werden Daten unter Einhaltung dieser Regeln veröffentlicht, so sind diese derart maschinenlesbar, dass neben der Möglichkeit der Darstellung auch deren Bedeutung, sofern diese explizit definiert wurde, maschinell verarbeitet werden kann.

Seit dem Aufkommen der Idee von Linked Data hat das Konzept vermehrt und noch immer zunehmend Anwendung gefunden. Abbildung 2.2 zeigt dies in einem Vergleich von Visualisierungen über Datensätze auf, die nach dem Linked Data Konzept veröffentlicht wurden.



(a) 19.09.2007, 12 Datensätze



(b) 01.05.2011, 295 Datensätze

Abbildung 2.2: Darstellung der Entwicklung von Linked Data anhand der Jahre 2007 (a) und 2011 (b), “Linking Open Data cloud diagram”, von Richard Cyganiak und Anja Jentzsch.
<http://lod-cloud.net/>

2.1.2 Semantic MediaWiki (SMW)

Dieser Abschnitt beschreibt die Software *Semantic MediaWiki*, welche für die Realisierung des Lösungsansatzes relevant ist, der in dieser Arbeit entwickelt wurde. Dazu wird zuerst die Software *MediaWiki* beschrieben, da diese die Basis von Semantic MediaWiki bildet.

MediaWiki

Ein *Wiki* ist eine Website, die nicht nur die Möglichkeit bietet deren Inhalte darzustellen, sondern auch diese auch kollaborativ zu bearbeiten [TCC04]. Die erste Realisierung eines Wikis geht auf Ward Cunningham zurück, der 1994 die Software *WikiWikiWeb* als ein Werkzeug für Wissensmanagement zur Verwaltung von Informationen über Entwurfsmuster zum Betrieb des *Portland Pattern Repository* implementierte [LC01].

Nach Ebersbach et al. [EGHW08] lässt sich ein Wiki durch folgende Eigenschaften charakterisieren:

Nichtlineare Hypertextstruktur Mithilfe von Wikis können Texte mit Hyperlinks versehen werden. Durch die damit verbundenen Querverweise unter den Hypertexten kann ein Netz aus Hypertexten gebildet werden.

Einfacher und weitgehender Zugriff Wikis sollen einen hohen Grad an Benutzbarkeit aufweisen. Auch Benutzer mit keinem oder nur geringem technischen Hintergrundwissen sollen die Inhalte eines Wikis bearbeiten können.

Keine Client-Software Ein Wiki kann allein mithilfe eines Browsers genutzt werden. Es ist keine weitere clientseitige Software nötig.

Soziale Prozesse im Vordergrund Organisatorische und inhaltliche werden vorrangig vor technischen Aspekten bei der Nutzung eines Wikis durch dessen Benutzergemeinschaft behandelt.

Da bei Wikis soziale Prozesse im Vordergrund stehen, hängt deren Einsatz von der Benutzergemeinschaft ab. Wikis können entweder als Werkzeuge von geschlossenen Arbeitsgruppen genutzt werden, oder sich über das Internet an alle richten [EGHW08].

Es existieren verschiedene Implementierungen von Wiki Software¹. *MediaWiki*² ist eine freie, quelloffene Software, welche ursprünglich zum Betrieb und der Verwaltung von Wikipedia erstellt wurde, inzwischen aber auch in

¹Das erste Wiki WikiWikiWeb von Ward Cunningham enthält eine Liste von Wiki Software, <http://c2.com/cgi/wiki?WikiEngines> (01.07.2014)

²<https://www.mediawiki.org/wiki/MediaWiki>

vielen anderen Projekten verwendet wird und auch die Basis der MediaWiki Erweiterung *Semantic MediaWiki* bildet, die im nächsten Abschnitt beschrieben wird.

Zur Vereinfachung der Bearbeitung und Verwaltung von Inhalten stellt die MediaWiki Software unterschiedliche Mittel bereit. Ein wichtiges Element dabei, stellt die MediaWiki Markup Sprache dar, mit deren Hilfe Inhalte formatiert und mit Hyperlinks versehen werden können. Die Spezifikation eines solchen Querverweises, die für die Semantic MediaWiki Erweiterung von Bedeutung ist, besteht mindestens aus dem Namen einer Seite. Beispielsweise können Hyperlinks auf Seiten innerhalb einer MediaWiki Instanz wie folgt dargestellt werden:

```
'''Heidelberg''' ist eine Stadt in [[Deutschland]].
```

Die Klammerung in dreifachen Anführungszeichen ist die Syntax zum Formatieren von Text in fett gedruckter Darstellung. Die Klammerung von Ausdrücken in doppelten, eckigen Klammern stellt den Querverweis dar. Hyperlinks auf externe Seiten können auch erstellt werden:

```
Weitere Informationen zu [http://mediawiki.org Mediawiki].
```

Ähnlich dazu können Seiten auch kategorisiert werden. Beispielsweise könnte eine Seite über die Stadt Heidelberg durch folgenden Ausdruck einer Kategorie zugeordnet werden:

```
[[Category:Stadt]]
```

Semantic MediaWiki

In diesem Abschnitt wird die MediaWiki Erweiterung *Semantic MediaWiki* beschrieben. Dazu werden im wesentlichen relevante Aspekte aus der Publikation [VKV⁺06] angeführt. Für weitere Informationen zu Semantic MediaWiki sei der Leser daher auf die Lektüre jener Publikation verwiesen.

Durch die Erweiterung *Semantic MediaWiki* wird MediaWiki um Funktionen ergänzt, die in Hinsicht auf die Semantic Web Bestrebungen sinnvoll sind. Der erste derartige Ansatz, welcher auch als Vorläufer von Semantic MediaWiki betrachtet werden kann, ist *Platypus Wiki*, das 2004 vorgestellt wurde, auf WikiWikiWeb basiert und bereits das RDF Modell und das Vokabular von RDFS und OWL verwendete, um Metadaten und Relationen zwischen Wiki Seiten zu repräsentieren [TCC04].

Das Projekt Semantic MediaWiki wurde mit dem Ziel gegründet, eine implementierte Erweiterung für MediaWiki zu schaffen, die in naher Zukunft in Wikipedia integriert werden sollte und die es ermöglichen sollte, wichtige Teile des in Wikipedia verfügbaren Wissens mit dem geringst möglichem Aufwand zur Verarbeitung durch Maschinen verfügbar zu machen.

Die wesentlichen Beiträge in Hinsicht auf die Semantic Web Bestrebungen

sind die Möglichkeiten durch Semantic MediaWiki Daten in Artikeln explizit als solche zu markieren, sowie diese mit weiteren Metadaten zu versehen und die Möglichkeiten diese Daten abzufragen, wodurch die maschinelle Verarbeitung ermöglicht wird.

Im folgenden wird näher beschrieben, wie die zuvor genannten Funktionen realisiert wurden und weshalb diese für die Integration von EM in ViFUn von Bedeutung sind. Die wichtigsten Elemente, welche durch Semantic MediaWiki eingeführt wurden, um semantische Daten in MediaWiki Inhalten zu annotieren, sind nach [VKV⁺06]:

Klassifizierung durch Kategorien Inhalte können durch Kategorien klassifiziert werden. Durch die Klassifizierung kann der Typ eines Artikels bestimmt werden, wodurch sich mit der Definition weiterer Axiome semantische Relationen zwischen den Artikeln herstellen lassen, wie bspw. deren Ähnlichkeit.

Die Angabe des Typs eines Artikels bzw. einer Entität, die durch einen Artikel repräsentiert wird, oder auch semantische Relationen - wie insbesondere die Ähnlichkeit - sind Voraussetzungen zur Anwendung vieler EM Ansätze.

Links mit Typenangabe Semantic MediaWiki bietet die Möglichkeit an, Links zwischen Artikeln mit einem Typ zu versehen. So lässt sich zum Beispiel ein Link mit Typangabe in Wiki Markup wie folgt angeben:

```
'''Heidelberg''' ist eine Stadt in [[Stadt in::Deutschland]]
```

Der Typ eines Links entspricht im Modell eines semantischen Netzes, das als Graph repräsentiert ist, der Art einer Relation zwischen zwei Knoten. Die Bedeutung einer Relation zwischen zwei Artikeln kann durch die Angabe des Typs derselben der maschinellen Verarbeitung zugänglich gemacht werden, wie bspw. durch die Angabe dass sich zwei Artikel auf die selbe Identität beziehen, wie es durch das Attribut *owl:sameAs* der Sprache OWL oder *skos:exactMatch* beim SKOS Vokabular definiert ist.

Relationen zwischen Artikeln bzw. der durch diese repräsentierten Entitäten sind wie zuvor erwähnt für viele EM Ansätze von Bedeutung. Sie sind überdies eine Voraussetzung für die Anwendung von *kontextbasierten* EM Ansätzen. Siehe dazu Abschnitt 2.3.4.

Attribute Neben der Klassifizierung und der Möglichkeit semantische Relationen zu anderen Artikeln herzustellen, können mit Semantic MediaWiki den Artikeln außerdem Attribute hinzugefügt werden. Dadurch lassen sich einfache Eigenschaften spezifizieren die im Zusammenhang mit dem Inhalt stehen, wie bspw. das Geburtsdatum bei einem Artikel über eine Person. Daher ähneln Attribute Links mit Typenangabe,

wobei jedoch eine Relation zwischen einem Artikel und einem Datum hergestellt wird.

Attribute werden im Wiki Markup wie folgt notiert:

`[[Attributname::Attributwert]]`

Soll einem Attribut zusätzlich ein Typ zugewiesen werden, so muss diesem Attribut selbst das vordefinierte Attribut *hasType* bzw. *Datentyp* mit entsprechendem Wert (bswp. Datum, URL, Zahl) zugewiesen werden:

`[[Datentyp::Datentypbezeichnung]]`

Neben der Bedeutung für die Semantic Web Bestrebungen ist die explizite Angabe von Daten in Form von Attributen eine Voraussetzung für die Verwendung *attributbasierter* EM Ansätze. Siehe dazu Abschnitt 2.3.4.

Die offenen W3C Standards RDF [MMM⁺04], XSD [FW04], RDFS [BG04] und OWL [WMS04] werden von Semantic MediaWiki verwendet, um die durch Menschen erstellten Informationen in Wikipedia maschinenlesbar zu machen. Dadurch ergeben sich außerdem neue Möglichkeiten, die Inhalte von Wikipedia durch existierende Semantic Web Software zu verarbeiten.

Es gab verschiedene weitere Herausforderungen, die durch Semantic MediaWiki zunächst nicht behandelt wurden. Zum Beispiel, dass nicht das gesamte Spektrum der Möglichkeiten, die OWL zur expliziten Spezifikation von Wissen bietet, ausgereizt wird.

Weiter muss möglicherweise, um mit existierender Semantic Web Software auf großen Datensätzen - wie sie die Wikipedia darstellt - arbeiten zu können, der Graph, der das semantische Netz der Wikipedia repräsentiert, in kleinere Teilgraphen zerlegt werden, um die Menge an Daten zu reduzieren. Dies kann auch die Anzahl der Operationen, die bei der Anwendung von EM nötig ist, stark reduzieren. Dies würde eine Form von *Blocking* darstellen, was im Abschnitt 2.3.4 näher erläutert wird.

Abbildungen von Wikipedia auf existierende Wissensdatenbanken sind eine weitere Herausforderung. Dass die in Wikipedia gespeicherten Daten weder konsistent noch vollständig sind wird in der Publikation [VKV⁺06] als besondere Herausforderung genannt. Dies zeigt, dass es einen Bedarf an Tools wie das, welches im Rahmen dieser Arbeit entwickelt wurde, gibt.

Diese Herausforderungen sind auch bis zum Verfassen dieser Arbeit nicht abschließend geklärt worden und sind noch immer Gegenstand der Forschung und Diskussion in Zusammenarbeit mit der MediaWiki bzw. Wikipedia Gemeinschaft im speziellen.

2.2 Virtuelle Forschungsumgebungen (ViFU)

Es gibt verschiedene Synonyme und Definitionen für das Konzept einer ViFU, wie *Science Gateways* [WD07], *Collaboratories* [KMW96], *Digital Libraries* [CCP⁺07], und *Inhabited Information Spaces* [SCF04], die sich unabhängig von einander in verschiedenen Disziplinen etabliert haben.

Diese Entwicklung ist unter anderem darin begründet, dass die Wissenschaft zunehmend global wird, wobei die Zusammenarbeit von Instituten verstärkt auch überregional und international stattfindet. Gleichzeitig nimmt die Vernetzung durch die globale Kollaboration zu, und die weltweiten Ausgaben für Forschung und die Entwicklung einfacherer und schnellerer Mittel zur Zusammenarbeit steigen, was für das Anhalten dieses Trends spricht [SS11].

Ein solches Mittel zur Zusammenarbeit stellen die so genannten *Virtuellen Forschungsumgebungen* dar. Nach einer populären Definition sind ViFUs als ein Framework zu sehen, in das Werkzeuge, Dienste und Ressourcen gesteckt werden können [Fra05].

Eine weitere bekannte Definition beschreibt ViFUs als flexible und sichere Arbeitsumgebungen im Internet, die darauf ausgerichtet sind, den Ansprüchen der modernen Wissenschaft zu genügen [CCP13].

ViFUs stellen die Mittel zur kollaborativen Bearbeitung von Forschungsfragen bereit. Insbesondere bedeutet dies, dass der Austausch von Informationen und die gemeinsame Bearbeitung konkreter Aufgaben zur Überwindung von bspw. zeitlichen oder örtlichen Hindernissen unterstützt wird.

"VREs are about enabling better collaboration." [Fra05].

2.2.1 Beispiele für ViFUs

Aus den im vorherigen Abschnitt 2.2 genannten Gründen sind zahlreiche ViFUs entstanden³, die speziell auf die Bedürfnisse der Wissenschaftler zugeschnitten sind, für die sie entwickelt wurden.

Dieser Abschnitt führt, um dies zu verdeutlichen zwei Beispiele von populären ViFUs an. Anschließend wird die ViFU *Semantic CorA*, anhand welcher die Erweiterung für *MediaWiki* bzw. *Semantic MediaWiki* realisiert wurde, die als ein Ergebnis im Rahmen dieser Arbeit entstanden ist, beschrieben.

*my*Experiment

Mit der ViFU *myExperiment* wurde das Ziel verfolgt, wissenschaftliche Arbeitsabläufe und Experimente, die in Computern durchgeführt werden leichter kollaborativ bearbeiten und teilen zu können [DRGS09].

³eine Liste von ViFUs ist unter <http://misc.jisc.ac.uk/vre/projects> zu finden

Dabei muss angemerkt werden, dass sich die Arbeit zwar zunächst nur auf Arbeitsabläufe konzentrierte, jedoch eine Menge von Arbeitsabläufen und mit ihnen in Verbindung stehende weitere Informationen ebenfalls in einem s.g. *Encapsulated^{my} Experiment Object* (EMO) gekapselt werden können. Auf diese Weise wird über die bloßen Arbeitsabläufe, Sammlungen derselben und die mit ihnen in Verbindung stehenden Informationen zum Konzept des s.g. *Research Object* abstrahiert.

Die ViFU ermöglicht es, diese Objektsammlungen in eine OAI Object Reuse and Exchange (OAI-OR) Repräsentation zu exportieren, was die Kompatibilität mit existierenden Diensten erhält. Beim Entwurf von *myExperiment* wurden Linked Data Praktiken angewendet [DRGS09].

Das Projekt wird als Social Media Dienst in Form einer Website⁴ seit dem Jahr 2007 für und in Zusammenarbeit mit der wissenschaftlichen Gemeinschaft entwickelt und umfasste zum Zeitpunkt des Verfassens dieser Arbeit 3317 Arbeitsabläufe und knapp 10000 Benutzer.

Neben den Möglichkeiten Arbeitsabläufe zu erstellen und zu teilen, bietet die Website inzwischen auch viele weitere Funktionen und Dienste an, wie die Visualisierung der Arbeitsabläufe und Werkzeuge um Arbeitsabläufe auszuführen und zu modifizieren. So können bspw. Experimente leicht reproduzierbar gemacht werden.

nanoHUB

*nanoHUB*⁵ “Online simulation and more for nanotechnology” ist eine ViFU mit der Ausrichtung auf Nanowissenschaften und Nanotechnologie.

Zum Zeitpunkt der Erstellung dieser Arbeit umfasste diese ViFU über 330 Simulationswerkzeuge und bietet ihren Benutzern außerdem Möglichkeiten zur Weiterbildung bspw. durch Präsentationen, Kurse und Podcasts an.

Abbildung 2.3 zeigt die Auswahl von Simulationswerkzeugen, die über die nanoHUB Website aufgerufen werden können.

⁴ <http://www.myexperiment.org/>

⁵ <https://nanoHUB.org/>

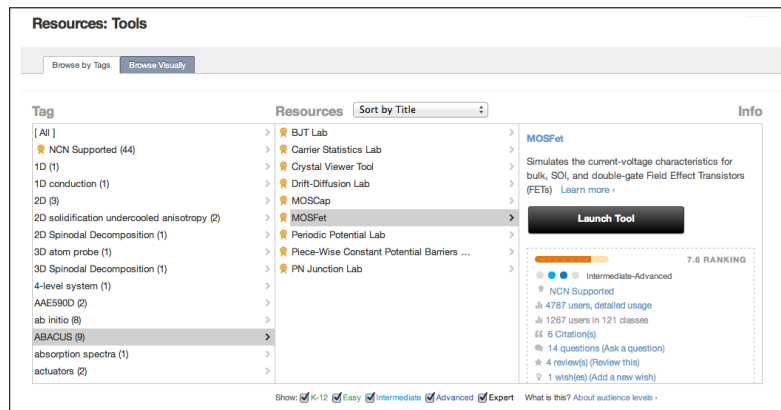


Abbildung 2.3: Die Übersicht und Auswahlmöglichkeit der in nanoHUB integrierten Simulationswerkzeuge mit Beschreibung und Informationen über deren Verwendung durch Benutzer.

Semantic CorA

Bibliotheken stellen ihre Inhalte zunehmend in digitaler Form bereit und bieten dadurch auch Möglichkeiten an, diese in ViFUs zu verwenden. Als Beispiel sei hier die Deutsche Nationalbibliothek⁶ genannt, die auch eine Informationsquelle der Forscher ist, die Semantic CorA verwenden.

In dem Ansatz *Semantic Collaborative Corpora Analysis*⁷ (Semantic CorA) wird eine ViFU mit Semantic Web Technologien vereint, mit dem Ziel die wissenschaftliche und bibliothekarische Praxis besser aufeinander auszurichten [SVRV11, SER12, SER13].

Dies wurde am Beispiel eines Integrationsprozesses für bibliografische Daten im Rahmen eines konkreten Forschungsprojektes aus dem Bereich der Historischen Bildungsforschung durchgeführt. Ein Teil der Projektarbeit stützt sich auf bibliografische Metadaten, die zur Erhebung statistischer Daten über die bibliografischen Datensätze selbst verwendet werden, von den Forschern zusätzlich mit weiteren Informationen angereichert und dann schließlich für qualitative Analysen herangezogen werden [SVRV11].

Das Projekt wird von der Deutschen Forschungsgemeinschaft (DFG) finanziert und ist durch eine Kooperation zwischen dem Deutschen Institut für Internationale Pädagogische Forschung (DIPF), dem Karlsruher Institut für Technologie (KIT), der Bibliothek für Bildungsgeschichtliche Forschung sowie der Georg-August-Universität Göttingen realisiert.

⁶ <http://www.dnb.de/>, Datensätze lassen sich über die Resultate einer Suchanfrage über <https://portal.dnb.de/> bereits als RDF/XML Repräsentation exportieren

⁷ <http://smw-cora.org/>

Semantic CorA basiert auf *MediaWiki*⁸ und *Semantic MediaWiki*⁹ (SMW), die zuvor im Abschnitt 2.1.2 vorgestellt wurden. Die Abbildung 2.4 zeigt die Startseite von Semantic CorA. Die Wiki Software wurde speziell an die Bedürfnisse der Forscher angepasst. Dazu zählt auch die Integration von Werkzeugen, wie der Import von Lexika und anderen Kategorien von Werkzeugen, wie in Abbildung 2.5 zu sehen ist.

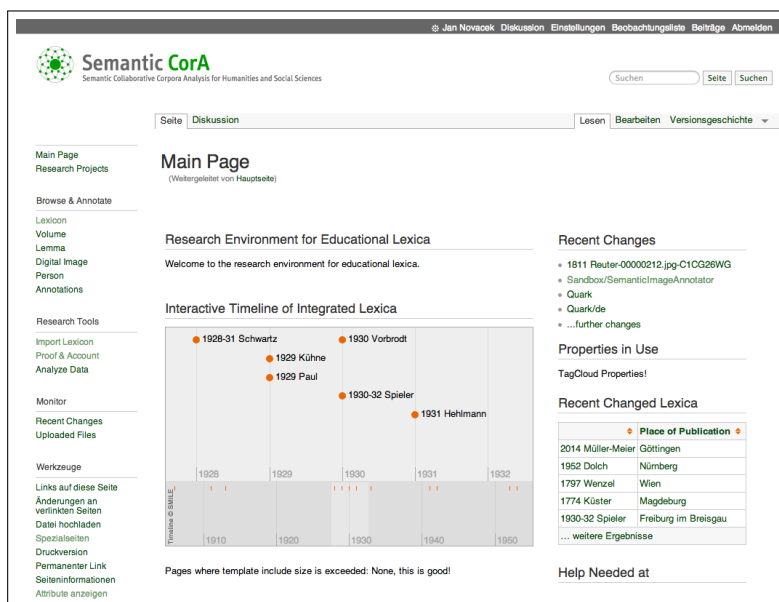


Abbildung 2.4: ViFU Semantic CorA Startseite.

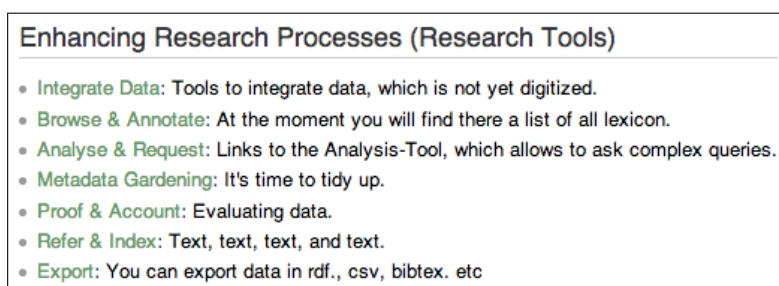


Abbildung 2.5: Übersicht über Kategorien von in Semantic CorA integrierten Werkzeugen, welche die Forscher bei der Bearbeitung der Forschungsfrage unterstützen.

⁸<http://www.mediawiki.org/>

⁹<http://semantic-mediawiki.org/>

2.3 Entity Matching (EM)

In diesem Abschnitt wird der Leser in das Thema Entity Matching eingeführt. Zunächst wird die geschichtliche Entwicklung von EM mit weiteren Hintergrundinformationen umrissen. Anschließend folgt eine Definition des EM Problems. Im Abschnitt 2.3.3 werden wichtige Anwendungen von EM vorgestellt, gefolgt von einem Überblick über EM Ansätze und wichtige Aspekte von EM im Abschnitt 2.3.4.

2.3.1 Hintergrund

Das EM Problem wurde zuerst 1959 von Newcombe et al. [NKAJ59] definiert als die beiden Teilprobleme, dass zum einen Daten, welche zur Identifikation einer Entität herangezogen werden fehlerhaft oder unvollständig sein können, was die Zuordnung erschwert und zum anderen in bestimmten Fällen, bei Duplikaten Ambiguitäten entstehen, die ebenfalls die Zuordnung erschweren. Das Problem wurde 10 Jahre später dann von Fellegi und Sunter formalisiert [FS69].

Es haben sich sehr viele gebräuchliche Synonyme für Entity Matching entwickelt, wie *Link Discovery*, *Entity Linking*, *Record Linkage*, *Entity Resolution*, *Deduplikation* etc. Diese Tatsache rührt unter anderem daher, dass die Anwendungen und Herausforderungen, von denen einige im folgenden Abschnitt 2.3.3 beschrieben werden, in verschiedenen wissenschaftlichen Fachrichtungen unabhängig behandelt wurden und sich daher in der Namensgebung unterscheiden. In dieser Arbeit wird allerdings stellvertretend nur der Ausdruck Entity Matching (EM) verwendet.

Deduplikation ist ein Beispiel für die Entwicklung eines solchen Synonyms, das vor dem Hintergrund der Entwicklung von Datenbanksystemen entstanden ist. Da Datenredundanz neben der Verschwendung von Speicherplatz außerdem noch zu verschiedenen Problemen wie der Handhabung von Änderungen, dem Entstehen von Inkonsistenzen und dem Fehlen einer zentralen, einheitlichen Datenhaltung führt, ist dies ein klassisches Problem bei Datenbanksystemen und wurde in der Vergangenheit bereits ausführlich behandelt. Ein Aspekt bei der Vermeidung von Redundanz ist die Erkennung von Duplikaten, woraus sich der Begriff *Deduplikation* entwickelt hat.

Der wesentliche Unterschied zwischen Deduplikation und EM ist, dass Deduplikation für gewöhnlich nur innerhalb einer Datenbank durchgeführt wird, wo alle Entitäten durch das gleiche Schema beschrieben werden [ES13].

Entity Matching und Linked Data

In jüngster Zeit wurden EM Ansätze entwickelt, die in Hinsicht auf Linked Data, die Zahl von Querverweisen zu erhöhen sollten [BdMNW12][VBGK09]. Das Ergebnis wäre die Gewährleistung der Linked Data Regel 4, die in Abschnitt 2.1.1 genannt wurde.

EM wird dabei verwendet, um Verküpfungen von Entitäten im Linked Data Web zu erzeugen. Meist werden diese Verküpfungen in Form von *owl:sameAs* Attributen der Sprache OWL spezifiziert, sie lassen sich aber auch in anderen Sprachen wie bspw. SKOS mit *skos:exactMatch*¹⁰ ausdrücken.

Wie Tabelle 2.6 zu entnehmen ist, stellt die Domäne der Life-Sciences den größten Anteil an Links im Linked Data Web. Den zweitgrößten Anteil machen Publikationen aus. Dies zeigt, dass es ein Potential für die Anwendung von Entity Matching zum Generieren von Links zwischen Entitäten im Kontext von Virtuellen Forschungsumgebungen und Linked Data gibt.

Domäne	Datensätze	Triples	%	Links	%
Medien	25	1,841,852,061	5.82 %	50,440,705	10.01 %
Geografie	31	6,145,532,484	19.43 %	35,812,328	7.11 %
Verwaltung	49	13,315,009,400	42.09 %	19,343,519	3.84 %
Publikationen	87	2,950,720,693	9.33 %	139,925,218	27.76 %
Übergreifend	41	4,184,635,715	13.23 %	63,183,065	12.54 %
Life-Sciences	41	3,036,336,004	9.60 %	191,844,090	38.06 %
Benutzerinhalt	20	134,127,413	0.42 %	3,449,143	0.68 %
	295	31,634,213,770		503,998,829	

Abbildung 2.6: Anzahl und Verteilung von LOD Datensätzen, Triples und Links pro Domäne, von <http://lod-cloud.net/state/>, Stand: 19.09.2011

¹⁰Während *owl:sameAs* dazu führt, dass Triples der verknüpften Ressourcen vereinigt werden, ist dies bei *skos:exactMatch* ausdrücklich nicht beabsichtigt [IS08].

2.3.2 Problem Definition

Dieser Abschnitt befasst sich mit der Definition des Entity Matching Problems. Die vorgestellten Definitionen basieren im wesentlichen auf der formalen Definition des Entity Matching Problems als Klassifikationsproblem von Fellegi und Sunter [FS69], die hier vereinfacht angeführt wird. Darüber hinaus wurde zudem die Definition einer Entität ergänzt.

Einfach ausgedrückt, ist es Gegenstand des Entity Matching (EM), Entitäten zu vergleichen und zu entscheiden, ob eine Entsprechung (*Match*) oder eine Nicht-Entsprechung (*Non-Match*) vorliegt oder ob diese Entscheidung aufgrund zu großer Unsicherheit nicht automatisch getroffen werden kann.

Das Problem kann auch nur auf die Entscheidung über eine Entsprechung von Entitäten aufgefasst werden [KR09].

Definition 1 (Entität). *Für ein Objekt $o \in O$ und dessen Eigenschaften $C_o \subseteq C$ seien Eigenschaftsdarstellungen $P_o \subseteq P$ durch die Abbildung $\rho : C \rightarrow P$ derart gegeben, dass $\forall \varrho \in P \exists c \in C : \rho(\varrho) = c$ und $\exists \varrho_1, \varrho_2 \in P : (\rho(\varrho_1) = \rho(\varrho_2) \nRightarrow \varrho_1 = \varrho_2)$, also ρ surjektiv und nicht injektiv ist.*

Analog dazu sei nun eine Objektdarstellung durch $\phi : O \rightarrow \Gamma$ gegeben, wobei $\Gamma = \mathcal{P}(\{\rho(\varrho) | \varrho \in P\}) = \{S | S \subseteq P\}$. Dann heißt $\gamma = \phi(o)$ Entität.

Definition 2 (Verknüpfungsvorschrift). *Für Objekte $o_1, o_2 \in O$ und Objektdarstellungen $\phi_1, \phi_2 \in \Phi$ sollen für die Entitäten $\gamma_1 = \phi_1(o_1), \gamma_2 = \phi_2(o_2)$ mit $\gamma_1 \in A, \gamma_2 \in B$ die Entscheidungen “Verknüpfung” (A_1), “Nicht-Verknüpfung” (A_2) und “Mögliche-Verknüpfung” (A_3) durch eine Verknüpfungsvorschrift getroffen werden.*

Definition 2.1 (Verknüpfung). *Die Entscheidung, ob $\phi_1^{-1}(\gamma_1) = \phi_2^{-1}(\gamma_2)$ ist, also ob $|\phi_1^{-1}(\gamma_1) \cap \phi_2^{-1}(\gamma_2)| = 1$ und $o_1 = o_2$, wird als “Verknüpfung” (A_1) bezeichnet. Die Wahrscheinlichkeit über den Fehler, dass also tatsächlich $\phi_1^{-1}(\gamma_1) \neq \phi_2^{-1}(\gamma_2)$, kann nach [FS69] definiert werden als*

$$\mu = \sum_{\delta \in \Delta} u(\gamma) P(A_1 | \gamma) \quad (2.1)$$

wobei δ eine Funktion über $A \times B$ ist, die als “Vergleichsvektor” bezeichnet wird und Δ die Menge aller möglichen Realisierung von δ ist, die “Vergleichsraum” genannt wird. $u(\gamma)$ ist die Wahrscheinlichkeit über die Realisierung von γ .

Definition 2.2 (Nicht-Verknüpfung). *Analog zu Definition 2.1 soll die Entscheidung, ob $\phi_1^{-1}(\gamma_1) \neq \phi_2^{-1}(\gamma_2)$, d.h. dass $|\phi_1^{-1}(\gamma_1) \cap \phi_2^{-1}(\gamma_2)| \neq 1$ und $o_1 \neq o_2$, als “Nicht-Verknüpfung” (A_2) bezeichnet werden. Die Wahrscheinlichkeit über den Fehler, dass $\phi_1^{-1}(\gamma_1) = \phi_2^{-1}(\gamma_2)$ soll definiert werden als*

$$\lambda = \sum_{\delta \in \Delta} m(\gamma) P(A_2 | \gamma) \quad (2.2)$$

Definition 2.3 (Mögliche Verknüpfung). *Die Entscheidung, ob keine Aussage über das Zutreffen von $\phi_1^{-1}(\gamma_1) \neq \phi_2^{-1}(\gamma_2)$ anhand eines festgelegten Fehler Schwellwertes gemacht werden kann, soll als “Mögliche Verknüpfung” (A_3) bezeichnet werden.*

Definition 2.4 (Verknüpfungsvorschrift). *Eine Verknüpfungsvorschrift (Linkage Rule) $L(\mu, \lambda, \Delta)$ weist die Wahrscheinlichkeiten $P(A_1|\gamma)$, $P(A_2|\gamma)$ und $P(A_3|\gamma)$ jeder Entität $\gamma \in \Gamma$ für die Werte (μ, λ) zu. Eine optimale Verknüpfungsvorschrift ist außerdem definiert als die Vorschrift, unter der $P(A_3)$ minimal ist.*

Definition 3 (Referenzverknüpfung). *Eine Referenzverknüpfung ist entweder eine Verknüpfung (A_1) oder eine Nicht-Verknüpfung (A_2), mit $\mu = \lambda = 0$.*

Anders ausgedrückt ist, wenn die Entscheidung Mögliche Verknüpfung (A_3) durch Vorwissen getroffen werden kann und bekannt ist, ob nur entweder eine Verknüpfung (A_1) oder Nicht-Verknüpfung (A_2) vorliegt, entweder (A_1) respektive (A_2) eine Referenzverknüpfung.

Definition 4 (Entity Matching). *Seien S_A und S_B Datenquellen, mit Mengen von Entitäten $A \in S_A$ und $B \in S_B$. Dann ist $A \times B$ die Vereinigung der beiden disjunkte Mengen*

$$M = \{(\gamma_1, \gamma_2); \gamma_1 \in A, \gamma_2 \in B, \phi_{\gamma_1}^{-1}(\gamma_1) = \phi_{\gamma_2}^{-1}(\gamma_2)\}$$

wobei $\phi_{\gamma_1}^{-1}$ die zu γ_1 und $\phi_{\gamma_2}^{-1}$ die zu γ_2 gehörende Objektdarstellung ist und

$$U = A \times B \setminus M$$

die Korrespondenz- und Nichtkorrespondenz-Menge genannt werden.

Das Entity Matching (EM) Problem besteht nun darin, mit Hilfe einer Verknüpfungsvorschrift $M \subseteq A \times B$ zu bestimmen.

2.3.3 Anwendungen und Herausforderungen

Zu den klassischen Anwendungen von Entity Matching zählt die Disambiguierung von gleichnamigen Entitäten - die s.g. *Named Entity Disambiguation* oder auch *Reference Disambiguation*, bswp. in [KM06]. Eine weitere klassische Anwendung ist die Disambiguierung ähnlicher Repräsentationen die sich auf die selbe Entität beziehen, was beispielsweise durch Fehler bei der Eingabe von Daten, z.B. durch Tippfehler, zustande kommt, oder auch durch fehlende Werte, wenn Vergleichsoperationen dadurch unterschiedliche Ergebnisse liefern.

Im Folgenden sollen einige wichtige Anwendungen von Entity Matching angeführt werden. Für einen umfangreichen Überblick über Entity Matching Anwendungen und Herausforderungen, sowie eine gute Einführung in das Thema sei der Leser zudem auf das Tutorial [GM12] verwiesen.

Datenbereinigung Das Erkennen und Beheben von Fehlern und Inkonsistenzen, mit dem Ziel, die Qualität der Daten zu steigern, ist Gegenstand der *Datenbereinigung*.

Nach [RD00] lassen sich die Probleme der Datenbereinigung sowohl nach der Anzahl der Datenquellen als auch nach Entitäts- und Schema-Ebene klassifizieren. Während sich Probleme, die auf der Schema-Ebene liegen durch Schema Evolution, Schema Integration und Schema Übersetzung lösen lassen, sind Probleme, die auf der Entitäts-Ebene liegen wie bspw. fehlerhafte Daten oder Inkonsistenzen nicht durch diese Verfahren zu lösen und daher auch Gegenstand der *Datenbereinigung*.

Such-Anfragen Suchanfragen können durch EM dadurch verbessert werden, dass Duplikate beseitigt werden oder die Suchergebnisse weitere Kandidaten basierend auf deren Ähnlichkeit zum eigentlichen Resultat enthalten.

Linked Data Bei der Umsetzung des Linked Data Konzeptes, welches zuvor im Abschnitt 2.1.1 beschrieben wurde, kann EM eingesetzt werden, um neben Gleichheit auch andere Beziehungen wie bspw. die Ähnlichkeit von Objekten zu inferieren.

Diese Bestimmung von *Co-Referenzen* oder *Equivalent URIs* zum selben Konzept oder der selben Entität stellt ein signifikantes Hindernis bei der Realisierung großer Web Anwendungen dar, findet aber in jüngster Zeit zunehmend Beachtung in vielen Forschungsgemeinschaften im Kontext des Semantic Web [GJM09].

Informations Extraktion Informations Extraktion (IE) erweitert Information Retrieval (IR) und informationsfilternde Systeme um die Funktion Daten aufzubereiten, um diese verfügbar zu machen [CL96]. Es kann als Front-End für hoch-präzise Informationsabfragen verwendet werden oder für die Weiterleitung von Texten, als erster Schritt von Systemen die zur Wissensentdeckung in Datenbanken nach Trends in enormen Mengen von Textdaten suchen oder auch als Eingabe für einen intelligenten Agenten, dessen Handlungen vom Verständnis textbasierter Informationen abhängen [Sod99].

EM kann in IE verwendet werden, um Ausdrücken die von einem Parser erzeugt wurden, Referenzen hinzuzufügen [CL96]. Diese Referenzen entsprechen einer Verknüpfung, bzw. Korrespondenz von Entitäten.

2.3.4 Ansätze und Aspekte

Neben der Differenzierung unterschiedlicher EM Ansätze werden in diesem Abschnitt auch wichtige Aspekte bei EM Verfahren hervorgehoben.

Es existieren verschiedene EM Ansätze, die wie zuvor erwähnt, innerhalb von unterschiedlichen wissenschaftlichen Disziplinen entwickelt wurden. Alleine im Rahmen dieser Arbeit konnten 33 verschiedene EM Ansätze identifiziert werden, die im Anhang E aufgelistet sind. Die Lösungswege verschiedener EM Systeme können sich unterscheiden. In diesem Abschnitt werden daher verschiedene EM Ansätze bzw. Unterscheidungsmerkmale vorgestellt.

Die Kriterien, welche zur Unterscheidung herangezogen werden, sind [KR09] entlehnt, worauf im Übrigen auch für einen guten Vergleich von EM Frameworks verwiesen sei. Die Kriterien wurden noch um das Unterscheidungsmerkmal der *Similarity Join* Algorithmen ergänzt, sowie die Unterscheidung nach dem Design der Anwendungsbereichsspezifität und um das Unterscheidungskriterium der Unterstützung bei der Spezifikation der Verknüpfungsvorschrift.

Eine Evaluation der Effektivität und Effizienz verschiedener EM Ansätze aus der wissenschaftlichen Gemeinschaft und von kommerziellen state-of-the-art Implementierungen ist in [KTR10] zu finden.

Art der Entitäten

Die Schwierigkeit der Definition sinnvoller Vergleichsoperationen für Entitäten zu definieren hängt von der Repräsentation der Entitäten ab, die verglichen werden sollen. Während Methoden für relationales EM auf der Annahme basieren, dass eine Entität durch ein Tupel repräsentiert wird und alle Attributwerte diese Entität beschreiben, erhöht sich im Vergleich dazu das EM bei komplex strukturierten Daten wie bei XML, was semi-strukturiert und hierarchisch organisiert ist. Weiterhin ist unklar, ob zwei Repräsentationen von unterschiedlicher Struktur sich auf die selbe Entität beziehen.

Blocking

Da das EM Problem nach Definition 4 grundsätzlich darin besteht, die Korrespondenzmenge an Entitäten zu bestimmen, welche eine Untermenge des Kartesischen Produkts aller Entitäten der Datensätze ist, sind bei großen Datensätzen *Blocking* Methoden nötig, zur Reduzierung der Anzahl an Vergleichsoperationen, um den hohen Rechenaufwand zu verringern. Gleichzeitig soll aber die Qualität der Ergebnisse erhalten bleiben. Blocking ist daher ein wichtiger erster Schritt aller skalierbarer EM Verfahren.

Die Anzahl der Vergleichsoperationen lässt sich reduzieren, indem die Datensätze in kleinere Blöcke (*Blocks*) zerlegt werden, auf denen dann die Vergleiche durchgeführt werden. Für die Vergleichsoperationen werden dann Blöcke

von Kandidaten ausgeschlossen, bei denen eine Verknüpfung sehr unwahrscheinlich ist oder ausgeschlossen werden kann. Die Unterteilung in Blöcke erfolgt traditionell durch einen Schlüssel, der sich für gewöhnlich aus einer Menge von Attributen (z.B. Nachname, Vorname, etc.) der Entitäten zusammensetzt [KR09]. Neben diesem Verfahren, das als *Standard Blocking* bezeichnet wird, wurden noch weitere Methoden entwickelt. Für einen Überblick über Blocking Methoden sei der Leser auf [BCC03] verwiesen.

Nach [KR09] lassen sich Blocking Methoden in *disjunkte* und *überlappende* Methoden einteilen. Disjunkte Methoden teilen Entitäten in Blöcke ohne Überschneidungen ein, d.h. jede Entität ist genau einem Block zugeteilt. Im Gegensatz dazu können bei überlappenden Blocking Methoden Entitäten auch mehreren Blöcken angehören.

Die Auswahl der geeigneten Attribute zur Einteilung der Entitäten in Blöcke, bzw. die Auswahl der geeigneten Blocking Strategie hängt stark von der Domäne ab. Eine suboptimale Blocking Methode kann dazu führen, dass zu viele unähnliche Paare von Entitäten ausgewählt werden, was sich auf die Performanz negativ auswirkt, oder was noch schlimmer ist, dass zu wenige wichtige ähnliche Paare von Entitäten ausgewählt werden, was zu einer Verringerung des positiven Vorhersagewertes (*precision*) führt. Die Auswahl kann deshalb nicht nur manuell, sondern auch semi-automatisch mittels Maschinellen Lernen erfolgen [BKM06].

Matcher

Ein Matcher trifft die Entscheidungen *Verknüpfung* (Definition 2.1), *Nicht-Verknüpfung* (Definition 2.2) und *Mögliche Verknüpfung* (Definition 2.3). Um dies zu erreichen, führt ein *Matcher* eine Ähnlichkeitsanalyse von Entitäten durch und misst dabei deren Ähnlichkeit durch die Verwendung von Ähnlichkeits- oder Distanzmaßen.

Matcher lassen sich nach den Kriterien, die sie zur Messung der Ähnlichkeit von Entitäten heranziehen, einteilen. Man unterscheidet zwischen Matchern, welche die Attribute von Entitäten zur Bestimmung der Ähnlichkeit heranziehen, den so genannten *attributbasierten Matchern* und Matchern, die andere Eigenschaften von Entitäten verwenden, wie bspw. Entitäten mit denen sie in Verbindung stehen. Da letztere sich auf den Kontext beziehen, in dem sich eine Entität befindet, werden derartige Matcher als *kontextbasierte Matcher* bezeichnet. Als Beispiel für kontextbasierte EM Verfahren sei R_{ELDC} [KMC05, KM06] genannt.

Zur Realisierung attributbasierter Matcher werden häufig Ähnlichkeits- oder Distanzmaße verwendet, die Zeichenketten vergleichen. Dazu zählen beispielsweise die Maße von Monge-Elkan [ME⁺96, ME97], die Metrik von Jaro [Jar89] mit dessen Erweiterung von Winkler [Win99], sowie Cohen mit dem

Vorschlag der Verwendung von TF-IDF für ein Distanzmaß [Coh00] oder die Methode von Fellegi und Sunter [FS69]. Ein ausführlicher Vergleich verschiedener Distanzmetriken ist in [CRF03] zu finden.

Matcher Kombinationen

Im Rahmen der Ähnlichkeitsanalyse können Matcher kombiniert werden. Liegen die Ähnlichkeitswerte der verschiedenen Matcher vor, so kann bspw. durch Gewichtung eine Entscheidung über die Ähnlichkeit zweier Entitäten getroffen werden. Nach Köpcke und Rahm [KR09] lassen sich Kombinationen von Matchern in 3 Gruppen einteilen:

Numerische Kombinationen Numerische Kombinationen von Matchern liefern die Ähnlichkeit zweier Entitäten als Ergebnis einer numerischen Funktion, deren Parameter die Ähnlichkeitswerte der einzelnen Matcher sind. Der resultierende Ähnlichkeitswert wird anschließend auf die Entscheidungen *Verknüpfung* (Definition 2.1), *Nicht-Verknüpfung* (Definition 2.2) und *Mögliche Verknüpfung* (Definition 2.3) abgebildet. Beispiele für numerische Kombinationen von Matchern sind in [Ngo12b] zu finden¹¹.

Regelbasierte Kombinationen In regelbasierten Kombinationen von Matchern wird die Entscheidung ob ein Entitätspaar der Korrespondenzmenge zugeordnet werden soll durch die logische Verknüpfung von Regeln getroffen. Als Beispiel für Kombinationen von Matchern seien die in [Ngo12b, Ngo12a] beschriebenen Ansätze genannt.

Eine Regel, bei der die Entscheidung über die Zuordnung zur Korrespondenzmenge nur durch einen Schwellwert getroffen wird und die nur numerische Matcher verwendet welche die Ähnlichkeit von Zeichenketten bestimmen, wird *Similarity Join* genannt und erlaubt in vielen Fällen eine hohe Performanz [AGK06].

Arbeitsablaufbasierte Kombinationen Die größte Flexibilität bieten *arbeitsablaufbasierte* (workflow-based) Kombinationen von Matchern. Sie ermöglichen es, Matcher frei zu kombinieren, wie bspw. unabhängig Vor- und Nachnamen zweier Entitäten zu vergleichen und anschließend die beiden Ähnlichkeitswerte zu einem Endergebnis zu kombinieren. Ein Beispiel für arbeitsablaufbasierte Kombination von Matchern ist *RiMOM* [LTLL09].

Die Spezifikation von Matchern, deren Kombinationen bzw. der Strategie in deren Rahmen Matcher zur Anwendung kommen (*Matching Strategie*) ist

¹¹Diese Veröffentlichung beschreibt außerdem den *HYPPO* (HYpersphere aPProximation algorithm) Algorithmus, der ausschließlich mit numerischen Attributwerten arbeitet.

selbst für Experten keine leichte Aufgabe, die außerdem sehr zeitaufwendig sein kann und großen Einfluss auf die Qualität der EM Ergebnisse hat [KR09].

Spezifikation der Verknüpfungsvorschrift

Wie im vorherigen Abschnitt erwähnt wurde, ist die Spezifikation der Matching Strategie selbst für Experten mitunter eine schwere und zeitaufwendige Aufgabe. Um dieses Problem zu lösen setzen einige EM Ansätze Maschinelles Lernen ein. Unter Verwendung von Trainingsdaten, die aus Entitäten der zu verknüpfenden Datensätze bestehen, soll dabei die Spezifikation der Verknüpfungsvorschrift (*Link Specification*) automatisch oder semi-automatisch gelernt werden.

EM Ansätze können daher unterteilt werden in Ansätze, bei denen die Verknüpfungsvorschrift manuell spezifiziert werden muss, solche bei denen die Verknüpfungsvorschrift semi-automatisch von Trainingsdaten gelernt wird und jene Ansätze, welche die Verknüpfungsvorschrift vollständig automatisch erlernen können. Letztere Ansätze stehen vermehrt im Fokus der wissenschaftlichen Gemeinschaft, da durch das Erlernen der Verknüpfungsvorschrift diese auf manuellem Wege oftmals schwierige und zeitaufwendige Aufgabe, stark vereinfacht werden kann.

Als Beispiel für einen Algorithmus zum Lernen von Verknüpfungsvorschriften der Überwachtes Lernen anwendet sei *GenLink* [IB12] angeführt. Beispiele für Algorithmen die Unüberwachtes Lernen verwenden sind der in [NdM12] beschriebene Algorithmus, sowie *RAVEN* [NLAH11], *EAGLE* [NL12] und *COLIBRI* [NSL14].

Auswahl von Trainingsdaten

Neben manueller, semi-automatischer und automatischer Spezifikation der Verknüpfungsvorschrift lässt sich auch die Art der Auswahl der Trainingsdaten, sprich wie die Entitäten die zum Lernen der Verknüpfungsspezifikation herangezogen werden, als weiteres Unterscheidungsmerkmal von EM Ansätzen verwenden.

Es kann zwischen automatischer und semi-automatischer Auswahl von Entitäten unterschieden werden. Bei der semi-automatischen Auswahl von Trainingsdaten werden Paare von Entitäten als Kandidaten ausgewählt, müssen dann aber manuell als *Verknüpfung* oder *Nicht-Verknüpfung* markiert werden. Es handelt sich also bei automatischer und semi-automatischer Auswahl von Trainingsdaten um unüberwachtes, respektive überwachtes Maschinelles Lernen.

State-of-the-art Ansatz zur semi-automatischen Auswahl von Entitäten, ist das so genannte *Active Learning*. Es basiert auf der Idee, die Anzahl der

Referenzverknüpfungen, die der Benutzer dem EM System liefern muss zu reduzieren, indem die am meisten informativen Kandidaten zur Kontrolle aktiv ausgewählt werden [Set10]. Beispiele für derartige Verfahren sind *ActiveGenLink* [IB13, Ise13], *EAGLE* [NL12] und *COALA* [NLC13]. Bei der Realisierung des Lösungsansatzes dieser Arbeit wurde *EAGLE* eingesetzt.

Similarity Join

Ein *Similarity Join* bestimmt alle Paare von Entitäten, deren Ähnlichkeit größer als ein bestimmter Schwellwert ist und ist daher ein wichtiger Bestandteil eines EM Systems. Beispielsweise verwendet das EM Framework *LIMES*, das bei der Realisierung des Lösungsansatzes dieser Arbeit verwendet wurde, den *PPJoin* Algorithmus [XWL⁺11] für diese Aufgabe.

Es kann zwischen linearen und parallelen *Similarity Join* Algorithmen unterschieden werden. Zu den linearen *Similarity Joins* gehören der Signaturbasierte Algorithmus *SSJoin/Part-Enum* [AGK06] unter Verwendung der Hammingdistanz, der Prefixfilter-basierte Algorithmus *AllPairs* [BMS07] welcher N-Gramme bildet, *Ed-Join* [XWL08] der *AllPairs* durch Reduzierung der Gram-Anzahl verbessert, *Trie-Join* [WFL10] welcher einen Prefixbaum für den *Similarity Join* verwendet, wobei auch nach Prefix gefiltert wird, *PPJoin* und *PPJoin+* [XWL⁺11] die *AllPairs* um Filterung nach Prefix, bzw. zusätzlich nach Suffix bei *PPJoin+* erweitern, *Pass-Join* [LDWF11] welcher N-Gramme bildet und der signaturbasierte Algorithmus *Fast-Join* [WLF14].

Zu den parallelen *Similarity Join* Algorithmen gehören beispielsweise ein Ansatz zu parallelen *Similarity Joins* von Mengen (*Set-Similarity-Join*, Menge von bspw. Zeichenketten) [VCL10], *Super-EGO* [Kal13], sowie Algorithmen die unter Verwendung der Burrows-Wheeler-Transformation eine effiziente approximative Suche auf Zeichenketten, sowie *Similarity Joins* mithilfe eines Prefixbaumes auf Multiprozessor- bzw. Multicore-Systemen ermöglichen [WYW13].

Grad der Anwendungsbereichsspezifität

Neben den bisher genannten Unterscheidungsmerkmalen lassen sich EM Ansätze auch danach unterscheiden, ob diese *domänenspezifisch* oder *universell* einsetzbar sind [NA11].

Domänenspezifische Ansätze zielen auf das Finden von Verknüpfungen zwischen Entitäten aus der selben Domäne ab. Beispiele hierfür sind der in [NUMDR09] beschriebene Ansatz und ORCHID [Ngo13]. Dagegen sind universelle Ansätze wie bspw. *RELDC* [KMC05][KM06] darauf ausgelegt, Entitäten auch aus unterschiedlichen Domänen zu verknüpfen.

Mit dem *Context Based Framework*¹² wurde auch der Ansatz verfolgt, mehrere domänenspezifische EM Frameworks zu kombinieren, um das Gesamtergebnis zu verbessern und den Grad der Unabhängigkeit von der Domäne zu erhöhen [CKM09].

2.4 Datenintegration

Die Integration von Daten identifizierter korrespondierender Entitäten in eine ViFU ist Teil der in Abschnitt 1.2 definierten Zielsetzung dieser Arbeit. Daher folgen eine Definition des Begriffes, sowie die Einordnung in den Kontext des Semantic Web und ViFUen.

2.4.1 Definition

In dieser Arbeit wird *Datenintegration* nach der Definition von Lenzerini [Len02] als das Problem aufgefasst, Benutzern Daten aus verschiedenen Quellen in einheitlicher Form bereitzustellen. Diese einheitliche Form kann als logische Relation in einem Datenbanksystem aufgefasst werden und wird daher auch als Sicht (*View*) bezeichnet.

Ansätze der Datenintegration lassen sich weiter nach [Len02] in zwei Gruppen unterteilen, die *Local-As-View* (LAV) [Ull97] und *Global-As-View* (GOV) [Hal01] genannt werden. Die beiden Ansätze unterscheiden sich in der Art, wie eine Sicht erzeugt wird.

2.4.2 Datenintegration und Linked Data

Das Linked Data Konzept, welches zuvor in Abschnitt 2.1.1 beschrieben wurde, unterscheidet sich von Datenintegration dadurch, dass Linked Data Objekte durch Links mit Typenangabe verknüpft, während bei Datenintegration externe Objekte oder Teile davon lokal dargestellt werden. Der Zusammenhang jedoch besteht darin, dass Inhalte miteinander verbunden werden.

Links mit Typenangabe, als Bausteine des Linked Data Web könnten dazu verwendet werden Datenintegration zu realisieren, indem die definierte Semantik des Links genutzt wird um die entsprechende Sicht zu erstellen.

Eine Anwendung von Datenintegration aus jüngerer Zeit im Zusammenhang mit Linked Data stellt die Big Data Integration (BDI) [DS13] dar. Ihr Gegenstand ist die Datenintegration im Angesicht aktueller Datenvolumina, die auch durch die Linked Data Entwicklung entstanden sind.

¹² Das Wort *Context* bezieht sich hier auf den Kontext der Anwendung des EM System, also die Domäne und nicht auf kontextbasierte Matcher.

2.4.3 Datenintegration und ViFUen

In ViFUen kann Datenintegration eingesetzt werden, um Benutzern eine einheitliche Sicht auf die Werkzeuge der ViFU und deren Ausgabedaten oder auf im Kontext der Forschungsfrage relevante Informationen zu liefern.

Um ViFUen für die Geisteswissenschaften zu erstellen, die nicht nur auf einen Bereich spezialisiert sind, sondern sich universell einsetzen lassen wurden Anwendungsfälle zu deren Verwendung untersucht, wobei insbesondere die Organisation und Integration der heterogenen Informationen auf denen diese Forschung basiert betrachtet wurde [BCH⁺10].

Kapitel 3

Architektur

In diesem Kapitel wird der Lösungsansatz, sowie die Architektur der Realisierung des Lösungsansatzes zur Erreichung der in Abschnitt 1.2 aufgestellten Ziele, vorgestellt.

Zunächst wird in Abschnitt 3.1 die Anforderungsanalyse dargestellt. Anschließend wird eine verwandte Arbeit in Abschnitt 3.2 vorgestellt. Der Lösungsansatz wird in Abschnitt 3.3 beschrieben, wobei zuerst Alternativen in Abschnitt 3.3.1 angeführt, dann die Entscheidung über die Auswahl in Abschnitt 3.3.2 und abschließend die Komponenten der Architektur in Abschnitt 3.3.3 vorgebracht werden.

3.1 Anforderungen

Dieser Abschnitt beschreibt, wie die Anforderungsanalyse durchgeführt wurde und welche Anforderungen erhoben wurden. Im ersten Abschnitt 3.1.1 wird beschrieben, wie die Anforderungsanalyse durchgeführt wurde. In den folgenden Abschnitten 3.1.3 und 3.1.2 werden dann explizit *nichtfunktionale* und *funktionale* Anforderungen aufgelistet. Zuletzt werden dann im Abschnitt refsectionConstraints überdies noch zusätzliche Randbedingungen festgelegt.

3.1.1 Anforderungsanalyse

Um die Anforderungen zu ermitteln, wurden zunächst Anwendungsfälle aufgestellt. Anschließend wurde ein Interview mit drei ForscherInnen durchgeführt, die im Umgang mit der ViFU vertraut waren, mit dem Ziel funktionale Anforderungen zu klären. Nichtfunktionale Anforderungen wurden danach von funktionalen Anforderungen abgeleitet oder sind der Literatur entnommen.

Von den Anwendungsfällen ausgehend wurden beispielhafte Anwendungsszenarien abgeleitet, welche anschließend durch die ForscherInnen bewertet

und kritisiert werden sollten. Die Fragen des Interviews wurden so gestaltet, dass sie durch die sich aus den Antworten und Anmerkungen ergebenden Arbeitsabläufen, die Ableitung von Anforderungen und die Ergänzung bzw. Anpassung der Anwendungsfälle ermöglichen.

Um die Durchführung dieses Interviews zu erleichtern, wurden dazu im Vorfeld kurze Beschreibungen von Anwendungsszenarien vorbereitet, welche Bestandteil der beiden wesentlichen, sich aus der Zielsetzung dieser Arbeit ergebenden, Anwendungsfälle sind. Anhand der Einschätzungen der Nützlichkeit von in den Anwendungsszenarien beschriebenen Funktionen durch die Forscher, sollten funktionale Anforderungen ermittelt und priorisiert werden. Die zur Diskussion verwendeten Anwendungsszenarien können Anhang D entnommen werden.

Anwendungsfälle

Im Folgenden werden die beiden Anwendungsfälle, die sich unmittelbar aus der Motivation und Zielsetzung dieser Arbeit ableiten lassen und durch das Interview im Rahmen der Anforderungsanalyse ergänzt wurden, nach dem Vorschlag der Notation von [ACB03] angeführt:

Anwendungsfall 1: Identifizierung externer Entitäten

BdViFU - Benutzer der Virtuellen Forschungsumgebung

Primärakteur BdViFU

Gelfungsbereich Virtuelle Forschungsumgebung Semantic CorA

Ebene Ziel auf Anwenderebene

Beschreibung Dieser Anwendungsfall umfasst alle Szenarien, die das Auffinden von korrespondierenden Entitäten in externen Datensätzen zum Ziel haben.

Vorbedingung Die ViFU enthält mindestens eine Entität.

Nachbedingung Die Anzahl der Entitäten wurde nicht verändert.

Standardablauf 1. BdViFU wählt Entitäten aus, die in externen Datensätzen identifiziert werden sollen.

2. BdViFU gibt den Speicherort des externen Datensatzes an (SPARQL Endpoint oder RDF Datei im lokalen Dateisystem).

3. BdViFU liefert dem EM System Referenzverknüpfungen.

4. BdViFU kontrolliert die durch das EM System gefundenen Verknüpfungen.

Erweiterungen 2a. Zugriff auf den externen Datensatz ist nicht möglich (Eingabedatei kann nicht gelesen werden oder SPARQL Endpoint ist nicht erreichbar).

2a1. Der Fehler wird dem BdViFU gemeldet.

2a2. BdViFU bricht den Anwendungsfall ab, oder versucht es erneut.

Variationen 3'. Referenzverknüpfungen werden vom EM System vorgeschlagen und müssen durch den BdViFU nur bestätigt oder abgelehnt werden.

4'. Verknüpfungen werden ohne Kontrolle in die ViFU integriert.

4''. Verknüpfungen werden nur bei ausgewählten Entitäten kontrolliert.

4'''. Es konnten keine Verknüpfungen gefunden werden.

Anwendungsfall 2: Anreicherung von Entitäten

BdViFU - Benutzer der Virtuellen Forschungsumgebung

Primärakteur BdViFU

Gelfungsbereich Virtuelle Forschungsumgebung Semantic CorA

Ebene Ziel auf Anwenderebene

Beschreibung In diesem Anwendungsfall werden alle Szenarien zusammengefasst, welche das Ziel haben Entitäten mit Daten anzureichern.

Verwendete Anwendungsfälle Anwendungsfall 1

Vorbedingung Mindestens eine Verknüpfung wurde durch Anwendungsfall 1 erstellt.

Invariante Die Anzahl der Entitäten bleibt unverändert.

Nachbedingung Den anzureichernden Entitäten wurden weitere Daten hinzugefügt.

Standardablauf 1. BdViFU spezifiziert eine Abbildung von dem Vokabular des externen Datensatzes in das der ViFU .
2. BdViFU wählt die Art der Daten aus, die integriert werden sollen.

Erweiterungen 2a. BdViFU gibt Regeln an, wie Daten bei der Integration modifiziert werden sollen (bspw. automatische Übersetzung).

Variationen 1a. Abbildung auf die Identität, zur identischen Übernahme von Attributen.

3.1.2 Funktionale Anforderungen

Alle funktionalen Anforderungen, die durch die Anforderungsanalyse, welche in Abschnitt 3.1.1 beschrieben wurde, erhoben werden konnten, werden in diesem Abschnitt angeführt.

- F1 Entity Matching** Die Software soll es dem Benutzer ermöglichen Korrespondenzen zwischen Entitäten der ViFU und denen, die ein externer Datensatz enthält, zu identifizieren.
- F2 Datensatzabstraktion** Die Software soll sowohl Datensätze in Form von RDF Dateien, als auch Datensätze, die über SPARQL Endpoints veröffentlicht werden, verarbeiten können.
- F3 Referenzverknüpfungen** Der Benutzer soll über die Software Referenzverknüpfungen zwischen Entitäten der ViFU und denen des externen Datensatzes anlegen und verwalten können.
- F4 Verwaltungsoperationen** Die Software soll Verwaltungsoperationen für Aufträge anbieten, wie bspw. Aufträge zu starten oder abzubrechen.
- F5 Ergebniskontrolle** Ergebnisse des EM Prozesses sollen durch den Benutzer kontrolliert bzw. korrigiert werden können.
- F6 Datenintegration** Liegen Ergebnisse aus dem EM Prozess vor, so soll der Benutzer die Software veranlassen können, Daten welche Teil externer Entitäten sind, den korrespondierenden Entitäten der ViFU hinzuzufügen.

3.1.3 Nichtfunktionale Anforderungen

Dieser Abschnitt beschreibt Anforderungen, die sich als Eigenschaften nicht nur auf einen Aspekt der Software beziehen oder solche, die sich aus technischen oder organisatorischen Gründen aus den zuvor beschriebenen Anforderungen ergeben.

- N1 Benutzbarkeit** Eine ViFU stellt, wie in Abschnitt 2.2 erklärt, die Mittel zur kollaborativen Bearbeitung von Forschungsfragen bereit. Benutzer einer ViFU müssen diese auch ohne technisches Hintergrundwissen oder lange Einarbeitung verwenden können, damit dies kein Hindernis bei der Bearbeitung einer Forschungsfrage darstellt.
Dies gilt aus dem selben Grund ebenfalls für Werkzeuge und Erweiterungen, welche in die ViFU integriert sind.
- N2 Kompatibilität** Nach [Fra05] sind kompatible Werkzeuge im Kontext von ViFUs ein wichtiger Aspekt der Wissenschaft im 21. Jahrhundert.
- N3 Wiederverwendbarkeit** Nebst Kompatibilität ist weiter nach [Fra05] auch die Wiederverwendbarkeit ein wichtiger Aspekt von Werkzeugen im Kontext von ViFUs .

N4 Wartbarkeit Die Software sollte einen möglichst hohen Grad an Wartbarkeit aufweisen. Das Entwicklungsmodell, welches im nächsten Kapitel in Abschnitt 4.1.1 näher beschrieben wird, erfordert stetige Weiterentwicklung und deshalb einen hohen Grad an Wartbarkeit.

N5 Skalierbarkeit Anwendbarkeit und Leistung der Software sollte nicht durch die Anzahl der Entitäten und deren Eigenschaften begrenzt werden, sodass die Bearbeitung diverser Forschungsfragen möglich ist.

3.1.4 Randbedingungen

Neben den funktionalen und nichtfunktionalen Anforderungen gab es weiterhin Randbedingungen, die für die Realisierung der Lösung beachtet werden mussten:

Terminierung Fertigstellung einer nutzbaren Version bis zum Abschluss dieser Arbeit. Bearbeitungszeitraum: 6 Monate.

Verfügbarkeit Für eine Veröffentlichung musste die Erweiterung für MediaWiki, sowie auch die Software, die diese verwendete mit der Lizenz von MediaWiki (GPLv2+) kompatibel sein.

3.2 Verwandte Arbeit

In diesem Abschnitt soll eine verwandte Arbeit, die während den Recherchen identifiziert werden konnte, kurz beschrieben, sowie Gemeinsamkeiten und Unterschiede zu dieser Arbeit herausgestellt werden.

Im Rahmen von dem in [BBEG10] vorgestellten Projekt, welches aus einer Kooperation zwischen verschiedenen Unternehmen und der Freien Universität Berlin hervorging, sollte das s.g. *Semantic MediaWiki Linked Data Environment* (SMW-LDE) erstellt werden. Der Kontext des Projekts ist der *Allen Brain Atlas*, der 2004 vom Allen Institut veröffentlicht wurde und Daten über Genexpressionen und Verbindungen zwischen Genen mit neuroanatomischen Informationen der Maus, des Menschen und nicht-menschlicher Primaten integriert [HNF⁺14].

Mit der Entwicklung von SMW-LDE wurden zwei wesentliche Ziele verfolgt: Die Datenintegration und Visualisierung einer großen Menge an relevanter neurogenetischer Daten und die Unterstützung kollaborativer Bearbeitung dieser Daten mittels Crowdsourcing im Stil von Wikis.

Die Autoren ordnen SMW-LDE nicht explizit in den Kontext von ViFUs ein. Nach den Definitionen, die in Abschnitt 2.2 angeführt wurden, kann SMW-LDE aber als Werkzeug, das bei der kollaborativen Bearbeitung von Forschungsfragen in einer ViFU zum Einsatz kommt, aufgefasst werden.

SMW-LDE basiert wie auch die ViFU Semantic CorA auf SMW. SMW-LDE

verwendet im Unterschied zu der Realisierung dieser Arbeit allerdings das kommerzielle Produkt SMW+, welches auf SMW basiert. Eine weitere Gemeinsamkeit mit dieser Arbeit ist die Integration von EM in SMW. Dies wurde bei SMW-LDE durch die Integration von Silk Server [IJB10] realisiert, was auch als Teil des Lösungsansatzes dieser Arbeit angestrebt wurde. Zu den Unterschieden zählt in Hinsicht auf die Zielsetzung, dass bei SMW-LDE die Datenintegration und die kollaborative Bearbeitung von Forschungsfragen im Vordergrund steht, während die Schwerpunkte dieser Arbeit sich auf die Anreicherung bereits existierender Daten in einer ViFU beziehen. Die Verwendung von EM unterscheidet sich zwischen SMW-LDE und dieser Arbeit auch dadurch, dass bei SMW-LDE EM mit statischen Verknüpfungsvorschriften eingesetzt wurde, da eine fixe Menge von Datensätzen integriert wurde. Nach der Zielsetzung dieser Arbeit soll aber EM so in eine ViFU integriert werden, dass dies Forscher bei der Identifikation von Entitäten in beliebigen externen Datensätzen unterstützt. Daraus ergibt sich auch der Fokus auf die Unterstützung der Forscher bei der Formulierung neuer Verknüpfungsvorschriften bzw. Matching Strategien, wie durch den Einsatz von überwachten und unüberwachten Maschinellen Lernenverfahren zum semi-automatischen Spezifizieren der Matching Strategie, wie zuvor in Abschnitt 2.3.4 beschrieben.

3.3 Lösungsansatz

Um der in Sektion 1.2 definierten Zielsetzung nachzukommen, muss die entsprechende Funktionalität, welche die Aspekte der Zielsetzung erfüllt, so bereitgestellt werden, dass diese von den Forschern, die mit der ViFU arbeiten, genutzt werden kann.

Grundsätzlich basiert der Lösungsansatz auf der Annahme, dass die ersten beiden Aspekte der Zielsetzung in geordneter Weise ablaufen. Das bedeutet, dass die semi-automatische Integration der Daten von identifizierten Entitäten als nächster Schritt nach der zuvor ausgeführten Identifikation erfolgt. Abbildung 3.1 stellt den Ablauf bei der semi-automatischen Anreicherung von Entitäten der ViFU dar.

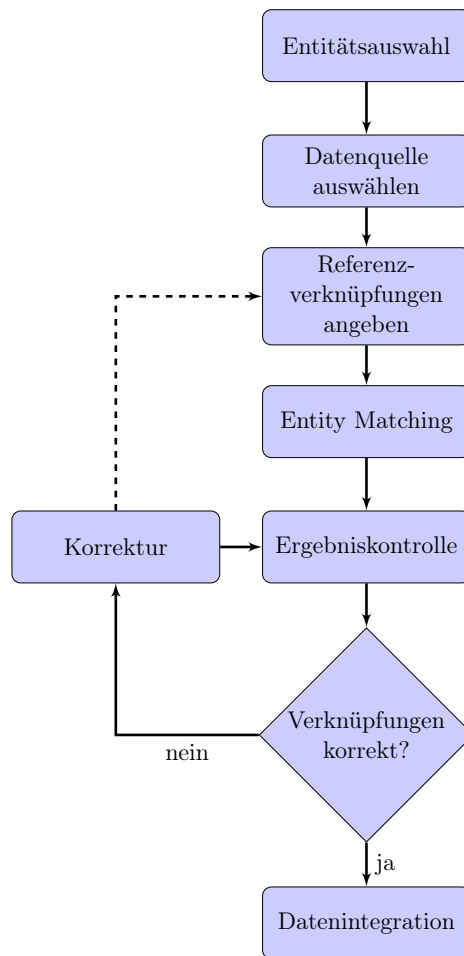


Abbildung 3.1: Ablauf bei der Anreicherung von Entitäten

Die Annahme dieses Ablaufs basiert auf der Tatsache, dass sofern existierenden Entitäten weitere Daten hinzugefügt werden sollen, zuerst bekannt sein muss, welche Entitäten in einem externen Datensatz zu diesen korrespondieren. Aus diesem Grund wurde der Aspekt der Identifikation von korrespondierenden Entitäten in einem externen Datensatz bei der Realisierung zuerst bearbeitet.

Zur Realisierung boten sich mehrere Vorgehensweisen an, die zuerst in Abschnitt 3.3.1 vorgestellt werden. Im darauf folgenden Abschnitt 3.3.2 wird außerdem eine Begründung über die gewählte Vorgehensweise bei der Realisierung gegeben, bevor dann in Abschnitt 3.3.3 die Komponenten und der Aufbau der Architektur der Realisierung beschrieben werden.

3.3.1 Alternativen

Bei der Realisierung standen, wie zuvor erwähnt, um die Anforderungen zu erfüllen, welche in den Abschnitten 3.1.3 und 3.1.2 festgelegt wurden, bei der Realisierung mehrere Optionen für die Herangehensweise zur Auswahl, die als grundlegende Architekturentscheidungen zu betrachten sind.

Dieser Abschnitt soll eine Übersicht darüber bieten, welche Optionen in Betracht gezogen wurden und anschließend die Entscheidung der Auswahl begründen.

Zunächst werden die Optionen dazu zusammen mit einer Beschreibung aufgelistet, anschließend verglichen und abschließend die Entscheidung über die Auswahl der entsprechenden Option begründet.

Es existieren noch weitere Möglichkeiten, wie beispielsweise verteilte EM Algorithmen zu implementieren, welche den Matching Prozess Benutzerseitig und verteilt durchführen. In Anbetracht der Realisierbarkeit, auch in Hinsicht auf die zuvor in Abschnitt 3.1.4 festgelegten Randbedingungen, wurden derartige Möglichkeiten nicht zur Auswahl herangezogen.

A1 Stand-Alone ausführen Zu vielen der in Anhang E betrachteten Ansätze stellten die Autoren unter anderem zur Reproduzierbarkeit zusätzlich Software bereit, die nur konfiguriert werden muss und dann eigenständig EM durchführt. Eine Möglichkeit EM in eine ViFU zu integrieren war daher, die ViFU um Konfigurations- und Steuermöglichkeiten für derartige Software zu erweitern. Die EM Software würde nach diesem Ansatz also direkt durch die ViFU Software ausgeführt werden. Durch die Kapselung der EM Methoden würde insgesamt die Modularität erhöht, was in Hinsicht auf die Software Evolution wünschenswert wäre. Nachteilig wären jedoch ständige Anpassungen, die sich durch Aktualisierungen der EM Software an den Konfigurations- und Steuermöglichkeiten ergeben, sowie die fehlende Anpassungsmöglichkeit der Software auf programmatrischer Ebene.

A2 Wiederverwendung vorhandener Implementierungen Im Unterschied zur vorherigen Möglichkeit, lassen sich auch nur die Implementierungen von Entity Matching Algorithmen wiederverwenden, da viele freie Autoren Open-Source Implementierungen anboten. Auf diese Weise ließen sich nur ausgesuchte Bestandteile von EM Software in eine ViFU integrieren und die Anpassung an die ViFU erleichtern. Im übrigen entspricht diese Option A1. Der wesentliche Nachteil im Vergleich zu A1 ist der höhere Aufwand, sowohl bei der Implementierung wie auch später bei der Wartung.

A3 Webservice Anders als bei den beiden vorangehenden Optionen A1 und A2, könnten EM Algorithmen oder lauffähige Software wiederverwendet und als eigenständiger Webservice bereitgestellt werden. Da-

durch entstünde neben der Integration von EM in die ViFU auch die Möglichkeit EM für andere Anwendungen bereitzustellen, um diese bspw. in eine Serviceorientierte Architektur zu integrieren. Dieser Ansatz bietet hohe Modularität durch die Trennung des Matching Back-Ends von der Steuerung über das Front-End in der ViFU, bringt aber den Nachteil mit sich, dass sich durch die Notwendigkeit der Installation eines separaten Webservices eine größere Abhängigkeit ergibt. Zudem ist der Aufwand höher als bei A2, da ein separater eigenständiger Dienst und die entsprechende Client Software implementiert und gewartet werden muss.

A4 Eigene Implementierung Es bestand weiterhin die Möglichkeit, eine eigene Implementierung eines EM Verfahrens zu erstellen. Da dies in der selben Programmiersprache wie die der ViFU geschehen kann, ließen sich so Abhängigkeiten auf ein Minimum reduzieren. Gleichzeitig bringt dieser Ansatz aber auch den meisten Aufwand mit sich und verwendet keine existierende Software wieder.

3.3.2 Architektur Auswahl

Im folgenden werden die im vorherigen Abschnitt 3.3.1 beschriebenen Ansätze weiter verglichen, um anschließend die Entscheidung über die Auswahl der Vorgehensweise zu begründen.

Die zum Vergleich herangezogenen Kriterien beziehen sich auf die in Abschnitt 3.1.3 erhobenen nichtfunktionalen Anforderungen. Um den Vergleich anzustellen wurden die Kriterien zusätzlich gewichtet, was im Übrigen die Priorität der entsprechenden Anforderung herausstellt.

Um den Vergleich anzustellen, wurden die Alternativen A1 bis A4 in der Tabelle 3.2 aufgetragen, die überdies eine Übersicht über die Alternativen bietet. Die Anzahl an + stellt die Bewertung der Attraktivität des entsprechenden Aspekts des zugehörigen Ansatzes dar. Wieß ein Aspekt keine Attraktivität auf oder war eine Bewertung nicht möglich, so wurde dies mit dem Symbol – gekennzeichnet.

Kriterium	Aufwand	Wiederverwendung	Skalierbarkeit	Wartbarkeit	Gesamt
Gewichtung	30%	20%	20%	30%	
A1	+	+	+	–	1,9
A2	+	+	+	–	1,1
A3	+	+	+	+	2,1
A4	–	–	+	+	1,1

Abbildung 3.2: Vergleich verschiedener Integrationsoptionen für Entity Matching in die Virtuelle Forschungsumgebung Semantic CorA.

Aus dem Vergleich geht hervor, dass die Alternativen A1 *Stand-Alone ausführen* und A3 *Webservice* die höchste Attraktivität besitzen. Bei näherer Betrachtung unterscheiden sich die beiden Ansätze in deren Wartbarkeit und dem zur Implementierung nötigen Aufwand. Unter Berücksichtigung der in Abschnitt 3.1.4 festgelegten Randbedingungen, wurde schließlich die Alternative A1 *Stand-Alone ausführen* ausgewählt.

3.3.3 Komponenten und Aufbau der Architektur

Dieser Abschnitt soll die Architektur der Lösung beschreiben, wozu ein Überblick über die Architektur gegeben wird und deren Komponenten erläutert werden. Eine Komponente bezeichnet in dieser Arbeit ein lose abgegrenztes Element der Architektur, das eine definierte Aufgabe erfüllt.

Zunächst erfolgt eine Beschreibung der Architektur auf einer abstrakten Ebene. Nachfolgend werden dann deren Komponenten genauer beschrieben, sowie die Interaktionen der Komponenten veranschaulicht. Der Detailgrad nimmt fortlaufend zu, bis hin zur Beschreibung der Implementierung im nächsten Kapitel 4.

Architektur Überblick

Für die grundlegende Herangehensweise beim Entwurf der Architektur der Realisierung wurde Alternative A1 *Stand-Alone ausführen* ausgewählt, die in Abschnitt 3.3.1 beschrieben und deren Auswahl im vorherigen Abschnitt 3.3.2 begründet wurde. Dieser Abschnitt soll einen Überblick über die Architektur der Lösung geben. Im folgenden Abschnitt 3.3.3 werden die Komponenten dann detaillierter beschrieben.

Die Architektur der Erweiterung basiert auf Entwurfsmustern von Architekturen zum Veröffentlichen und Verarbeiten von Linked Data nach Heath und Bizer [HB11]. Während hier mit der Verarbeitung der Zugriff auf externe Datensätze gemeint ist, bezieht sich das Veröffentlichen in diesem Zusammenhang auf die Bereitstellung von Ergebnissen, die aus dem EM Prozess hervorgehen. Abbildung 3.3 zeigt den Vorschlag der Architektur zum Verarbeiten von Linked Data nach dem *Crawling Pattern*.

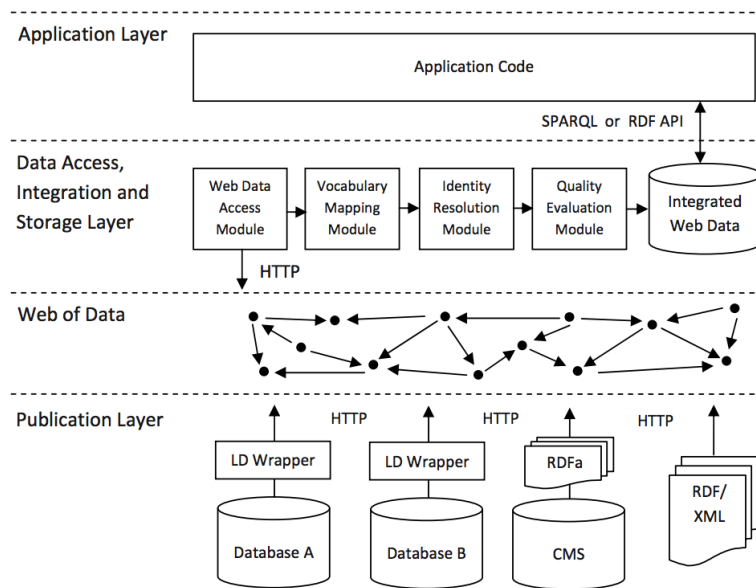


Abbildung 3.3: *Crawling pattern*, aus [HB11]

Neben der Verarbeitung von Linked Data wurden auch Entwurfsmuster für die Entwicklung der Architektur zum Veröffentlichen von Linked Data verwendet, die in Abbildung 3.4 dargestellt werden. Die Architektur des Lösungsansatzes dieser Arbeit enthält die Entwurfsmuster zum Veröffentlichen strukturierter Daten.

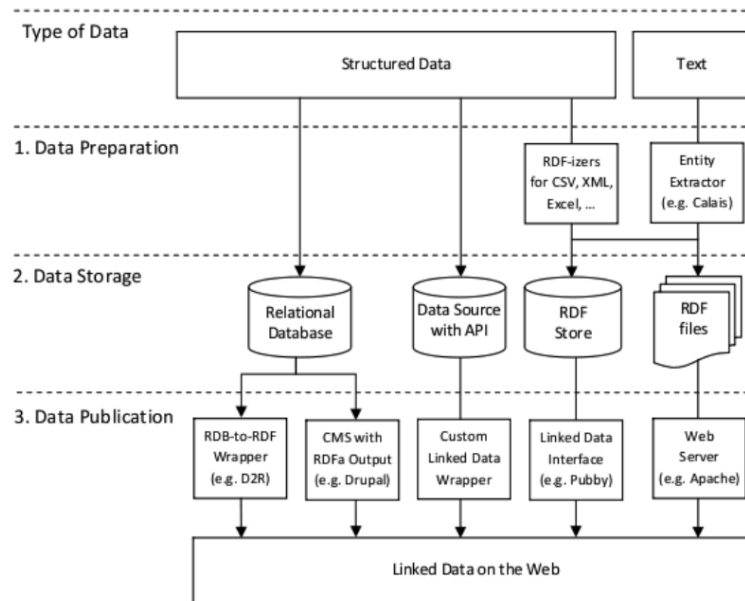


Abbildung 3.4: Alternative Arbeitsabläufe beim Veröffentlichen von Linked Data, aus [HB11]

Um die ViFU um Steuerungsmöglichkeiten für existierende EM Software zu ergänzen, werden Komponenten zur Verwaltung von Aufträgen und Datensätzen, zur Verwaltung der Auswahl an Entitäten und von Referenzverknüpfungen benötigt. Diese Teile der Architektur gewährleisten die funktionalen Anforderungen F1 bis inklusive F4, die in Abschnitt 3.1.2 festgelegt wurden. Weiter werden Teile der Architektur benötigt, welche die Anforderungen F5 *Ergebniskontrolle* und F6 *Datenintegration* erfüllen.

Die Funktionalität welche durch die Komponenten zur Verfügung gestellt wird, wird im nächsten Abschnitt 3.3.3 erläutert.

Komponenten der Architektur

Nach Ebersbach et al. [EGHW08] kennt Wiki-Technologie keine Unterscheidung zwischen Back-End und Front-End. Dies lässt sich auf eine benutzerzentrierte Sichtweise auf Wiki-Software zurückführen. In dieser Arbeit soll diese Unterscheidung aber dennoch durchgeführt werden. Alle Komponenten, die der Darstellung und Steuerung der Software aus Sicht des Benutzers dienen, werden dem Front-End zugeordnet. Die übrigen Komponenten, die keine Front-End Komponenten sind, werden dem Back-End zugeordnet. Abbildung 3.5 zeigt das Komponentendiagramm der Architektur.

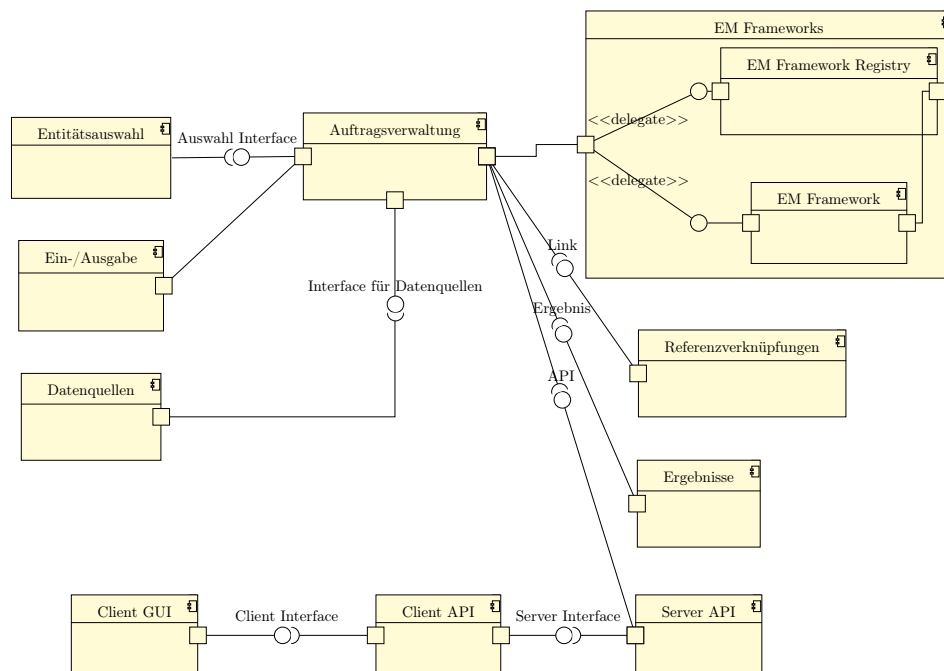


Abbildung 3.5: Komponentendiagramm der Architektur

In Hinsicht auf das in Abbildung 3.3 dargestellte *Crawling Pattern* sind die Komponenten der vorgestellten Lösung auf der Anwendungsebene und der Ebene für Datenzugriff, Integration und Speicherung angesiedelt. Im Folgenden werden nun alle Komponenten der Architektur beschrieben und deren Existenz durch die Reduzierung auf die entsprechenden Anforderungen begründet.

Entitätsauswahl Die Komponente zur Entitätsauswahl bietet Funktionen zur Verwaltung von Listen von Entitäten an, die angereichert werden sollen. So können Entitäten ausgewählt werden und diese Auswahl gespeichert und später erneut verwendet oder modifiziert werden. Diese Komponente ist nicht für die Erfüllung der funktionalen Anforderungen aus Abschnitt 3.1.2 notwendig, sondern in Hinsicht auf die in Abschnitt 3.1.3 festgelegte nichtfunktionale Anforderung N1 *Benutzbarkeit*.

Datenquellen Da Entitäten der ViFU mit Daten aus externen Datensätzen angereichert werden sollen, ist eine Komponente zur Angabe und ggf. Verarbeitung jener Datenquellen erforderlich. Die Komponente zu Datenquellen soll RDF Dateien verarbeiten können und SPARQL End-point URLs für die EM Software verwalten. Die Realisierung ist im nächsten Kapitel unter Abschnitt 4.2.2 beschrieben.

Referenzverknüpfungen Zu den funktionalen Anforderungen gehört auch, dass der Benutzer die Möglichkeit hat, der EM Software Referenzver-

knüpfungen von Entitäten zu liefern oder diese zu bestätigen. Dies ist deshalb vorgesehen, da EM Software eingesetzt werden sollte, die Maschinelles Lernen verwendet, um die Verknüpfungsvorschrift (semi-)automatisch von Trainingsdaten zu erlernen, was etwas näher in Abschnitt 2.3.4 des Grundlagen Kapitels erklärt ist. Die Komponente für Referenzverknüpfungen dient also durch das maschinelle Lernen von Verknüpfungsvorschriften der Unterstützung des Benutzers und damit der Erfüllung von Anforderung N1 *Benutzbarkeit*.

EM Frameworks Eine Komponente, die EM Software wiederverwendet und steuert ist ein zentrales Element des Lösungsansatzes und zur Erfüllung der Anforderung F1 *Entity Matching* erforderlich. Informationen zur Realisierung dieser Komponente sind im nächsten Kapitel zur Implementierung unter Abschnitt 4.2.1 zu finden.

Auftragsverwaltung Die Auswahl von Entitäten, zugehörige Referenzverknüpfungen sowie Ergebnisse eines EM Prozesses werden zu einem Auftrag zusammengefasst. Durch die Komponente zur Auftragsverwaltung wird die Benutzbarkeit der Erweiterung erhöht, da diese Möglichkeiten zur Steuerung bietet. Außerdem können so verschiedene Arbeitsabläufe durch die Abbildung auf Aufträge parallel bearbeitet werden. Es wird zudem die Unterbrechung und Fortsetzung der Arbeit an Aufträgen ermöglicht oder bswp. im Nachhinein weitere Entitäten zur Auswahl hinzuzufügen.

Ergebnisse Liegen Ergebnisse eines EM Prozesses vor, so wird eine Komponente benötigt, die diese dem Benutzer zur Verfügung stellt. In dieser Komponente werden auch die Datenintegrationsfunktionen realisiert. Die Funktionen müssen so entworfen werden, dass zur Verfügung stehende Ergebnisse durch den Benutzer kontrolliert und ggf. korrigiert werden können. Diese Komponente dient also der Erfüllung der drei funktionalen Anforderungen F1 *Entity Matching*, F5 *Ergebniskontrolle* und F6 *Datenintegration*.

Ein-/Ausgabe Die Anforderungen F2 *Datensatzabstraktion* und N2 *Kompatibilität* erfordern eine Komponente, welche die Verarbeitung von unterschiedlichen Ein- und Ausgabeformaten ermöglicht. Um die Kompatibilität zu erhöhen, sollten außerdem offene Standards eingesetzt werden.

Hintergrundaufgaben In Abhängigkeit von der Größe der Datensätze kann die Verarbeitung durch den EM Prozess viel Zeit in Anspruch nehmen. Aus diesem Grund wird eine Komponente benötigt, die Aufträge im Hintergrund ausführen kann, sodass der Benutzer in der Zwischenzeit andere Aufgaben bearbeiten kann. Diese Komponente existiert also zur Einhaltung der Anforderung N1 *Benutzbarkeit*.

Server API Um die Darstellung von der Logik zu trennen, werden Server und Client Software separat entwickelt. Die Kommunikation erfolgt dann über eine serverseitige API, über welche die Erweiterung gesteuert werden kann. Um die Anforderungen N3 *Wiederverwendbarkeit* und N4 *Wartbarkeit* abzudecken, wird diese Schnittstelle modular in einer Komponente realisiert.

Client API Entsprechend zur Beschreibung der Server API Komponente wird auch bei der Client Software die Darstellung von der Logik getrennt. Bei der Logik des Clients wurde dann zusätzlich noch die API als eigene Komponente realisiert. Diese Unterteilung ist insbesondere der Erfüllung der Anforderung N3 *Wiederverwendbarkeit* zuträglich, da auf diese Weise eine Client Software geschaffen wird, die sich in andere Software integrieren lässt oder es ermöglicht, eine andere GUI zu entwickeln.

Client GUI Wie zuvor beschrieben, wird die Darstellung getrennt von der Logik entwickelt. Die Komponente in der die Darstellung des Clients realisiert wird, kapselt den Aufbau und die Funktionen aller benutzerseitigen Bedienelemente der Software.

Hilfswerkzeuge Alle Teile der Software, die keiner der zuvor angeführten Komponenten zugeordnet werden können oder aus entwurfstechnischen Gründen bei der Implementierung von diesen getrennt werden müssen, werden der Komponente zugeordnet, die alle Hilfswerkzeuge zur Realisierung anderer Komponenten enthält.

Bei der Betrachtung der vorgestellten Komponenten der Architektur wird deutlich, dass mehr Funktionalität bereitgestellt werden soll, als für die Erfüllung der in Abschnitt 3.1.2 festgelegten funktionalen Anforderungen erforderlich wäre. Dieser scheinbare Umstand ist darin begründet, dass die Funktionen, die nicht eindeutig den funktionalen Anforderungen zugeordnet werden können von nichtfunktionalen Anforderungen abgeleitet wurden. Als Beispiel sei hierfür die Möglichkeit zur Auswahl von Entitäten als Funktion zur Erfüllung der nichtfunktionalen Anforderung N1 *Benutzbarkeit* genannt. Weitere Erläuterungen zum Zweck dieser Funktion können der Beschreibung der Entitätsauswahl-Komponente im folgenden Abschnitt entnommen werden.

Kapitel 4

Implementierung

In diesem Kapitel wird beschrieben, wie der im vorherigen Kapitel erläuterte Lösungsansatz implementiert wurde. Dazu wird als erstes in Abschnitt 4.1 ein Überblick über das Vorgehensmodell bei der Entwicklung, die verwendeten Technologien und die wesentlichen Herausforderungen der Implementierung geboten. Anschließend werden in den Abschnitten 4.2 und 4.3 die Implementierung der Back-End und Front-End Komponenten genauer beschrieben und Probleme bei der Entwicklung diskutiert. Die Definition von Back- und Front-End wurde bereits in Abschnitt 3.3.3 festgelegt.

4.1 Überblick

Um den in Abschnitt 3.1.1 erhobenen Anforderungen nachzukommen und diese in der Architektur zu realisieren, deren Auswahl in Abschnitt 3.3.2 begründet wurde, wurde eine Erweiterung für MediaWiki¹ unter Verwendung der SMW Erweiterung für MediaWiki erstellt.

Die Objektorientierte Programmierung wurde als Programmierparadigma bei der Entwicklung angewendet. Die grafische Benutzerschnittstelle der Front-End Komponente verfolgt zusätzlich das Paradigma der parallelen, ereignisorientierten Programmierung. Beim Entwurf der Software wurden außerdem verschiedene Prinzipien beachtet, die in [Mar03] definiert wurden. Dazu zählen *Single Responsibility Principle* (SRP), *Open-Closed Principle* (OCP), *Common Closure Principle* (CCP), *Liskov Substitution Principle* (LSP), *Interface Segregation Principle* (ISP) und *Dependency Inversion Principle* (DIP).

Insgesamt ist die Implementierung der MediaWiki Erweiterung auf 30 Quelltexte aufgeteilt, wobei 25 Quelltexte in PHP verfasst wurden, ein Quelltext

¹Unter http://www.mediawiki.org/wiki/Category:All_extensions registrierte Erweiterungen, die durch die MediaWiki Entwickler Gemeinschaft entstanden sind, umfassen zum Zeitpunkt des Verfassens dieses Dokuments 2210 Erweiterungen (abgerufen am 01.07.2014)

in Scala und die übrigen Quelltexte in HTML, CSS, JavaScript und SQL geschrieben wurden. Alle Quelltexte umfassten zusammen 4534 Zeilen, wobei der Anteil von PHP mit 3149 Zeilen (69,5 %) am größten ist.

4.1.1 Vorgehensmodell

In der Planungsphase wurde die Entwicklung in zwei Phasen eingeteilt, wobei die erste Phase aus Recherche und Einarbeitung in MediaWiki, SMW, PHP und das Schreiben von Erweiterungen für MediaWiki und SMW beinhaltete. Die zweite Phase, in welcher die Implementierung durchgeführt wurde, sah die Anwendung eines iterativen und inkrementellen Entwicklungsmodells vor. Darin war vorgesehen, die Benutzer der ViFU in den Entwicklungsprozess mit einzubeziehen, um die Anforderungen kontinuierlich zu verfeinern. Um die Implementierung der wissenschaftlichen Gemeinschaft zugänglich zu machen, sollte die MediaWiki Erweiterung als Open-Source Projekt veröffentlicht werden, was potentiell weitere Nutzer in den Entwicklungsprozess hätte integrieren können. Dies war maßgeblicher Grund für die Wahl des Entwicklungsmodells, das abgesehen von der Fertigstellung eines nutzbaren Prototyps innerhalb des Bearbeitungszeitraums dieser Arbeit, ohne zeitliche Begrenzung entworfen wurde.

In jeder Iteration der zweiten Entwicklungsphase sollten vier Schritte durchgeführt werden, die dann zusammen ein Inkrement zum Ergebnis haben sollten. Um das Entwicklungsmodell zu beschreiben, werden die vier Schritte dieser Phase genannt:

1. Programmierung
 - (a) Feingranulares Ziel definieren
 - (b) Schritte identifizieren
 - (c) Entwurf
 - (d) Implementierung
 - (e) Test
2. Fortschritt dokumentieren
3. Ergebnisse evaluieren
4. Anforderungen anpassen

Zeitliche Rahmen für die einzelnen Schritte, Teilschritte und die gesamte Iteration wurden nicht festgelegt, um die Flexibilität zu erhalten und den Verwaltungsaufwand durch die Anwendung dieses Entwicklungsmodells zu verringern. Die Definition der feingranularen Ziele, die Identifizierung der notwendigen Schritte bei der Implementierung, der Entwurf, sowie die Dokumentation des Fortschritts und der Ergebnisse wurden pro Iteration in schriftlicher Form abgefasst.

4.1.2 Eingesetzte Technologien

Zu Beginn jeder Iteration der zweiten Entwicklungsphase wurden auch geeignete Technologien gesucht, um entsprechende Ziele umzusetzen. Für die Back-End Komponenten wurden PHP 5.3.10, Java 1.7.0 und Scala 2.11.0 verwendet, was im Abschnitt 4.2 näher beschrieben wird. Auf der Benutzerseite, bzw. dem Front-End wurde JavaScript zur Implementierung des Clients verwendet, was in Abschnitt 4.3 beschrieben wird. Darüber hinaus wurden spezielle Werkzeuge erstellt, wobei ebenfalls Scala eingesetzt wurde. Die speziellen Werkzeuge werden im letzten Abschnitt 4.4 beschrieben.

4.1.3 Herausforderungen der Implementierung

Zu den Herausforderungen bei der Implementierung zählte der Entwurf eines Entwicklungsmodells zur Einbindung von Benutzern der ViFU in die Entwicklung, um einen kontinuierlichen Entwicklungsprozess zu schaffen, mit dem Ziel die Software Anforderungen fortlaufend weiter zu verfeinern.

Eine weitere Herausforderungen war die Erfüllung der Anforderung N1 *Benutzbarkeit*, die in Abschnitt 3.1.3 aufgeführt ist, um die Forscher, welche die ViFU Semantic CorA verwenden, möglichst gut bei der Bearbeitung der in der Anforderungsanalyse ermittelten Anwendungsfälle 1 und 2, die in Abschnitt 3.1.1 festgehalten wurden, zu unterstützen. Neben der soeben genannten nichtfunktionalen Anforderung betraf dies zudem auch die funktionalen Anforderungen F4 *Verwaltungsoperationen* und F5 *Ergebniskontrolle*, die in Abschnitt 3.1.2 aufgestellt wurden, da diese über ausreichend viele Verwaltungsoperationen verfügen sollten.

Da existierende EM Software wiederverwendet werden sollte, ergab sich das Risiko dass diese nicht mehr gewartet wird und damit im Laufe der Zeit ihre Nützlichkeit verliert. Dieses Problem sollte durch die Abstraktion über EM Frameworks gelöst werden, wodurch mehrere EM Frameworks verwendet werden können. Die Abstraktion über EM Frameworks bietet darüber hinaus auch die Möglichkeit, verschiedene ER Ansätze zu kombinieren, um die Qualität der Ergebnisse zu verbessern und das Problem der Kontextabhängigkeit zu behandeln [CKM09].

4.2 Implementierung des Back-Ends

Zu den Back-End Komponenten der Software zählen alle Klassen zur serverseitigen Verwaltung der Datensätze, der Entitäts-Auswahlen, der Ergebnisse eines Entity Matchings, der Referenzverknüpfungen und der Aufträge. Außerdem sind auch Klassen zur serverseitigen Steuerung von EM Software Teil des Back-Ends. Weiter zählen Klassen zur Verwaltung von SPARQL Endpoints und Klassen die für die Ein-/Ausgabe in speziellen Formaten verwen-

det werden, zum Back-End. Klassen, welche die Funktionalität der serverseitigen API der Erweiterung implementieren, können sowohl dem Back-End wie auch dem Front-End zugeordnet werden, da diese eine von außerhalb des Systems nutzbare Schnittstelle bereitstellen.

Um die in Abschnitt 3.1.3 festgelegte Anforderung N2 *Kompatibilität* zu erfüllen, wurden offene Standards für Ein- und Ausgabedateiformate verwendet. Zu diesen zählen XML/RDF, sowie SPARQL zur Abfrage von RDF. Die Angabe von Referenzverknüpfungen und die Ausgabe von Ergebnissen eines EM Prozesses (*Alignment*) erfolgt im *Ontology Alignment Format*² (OAF). Die Spezifikation von Verknüpfungen von Entitäten kann sowohl Verknüpfungen zwischen Entitäten innerhalb einer Ontologie, wie auch Verknüpfungen zwischen mehreren unterschiedlichen Ontologien enthalten, was am Beispiel einer OAF Datei in Auszug Im Anhang B.1 gezeigt wird.

Neben Klassen zum Einlesen und für die Ausgabe von OAF Dateien wurde das Modell eines Alignments von OAF auch auf Klassen abgebildet, die intern zur Repräsentation von Alignments genutzt werden. Auf diese Weise sollte die Kompatibilität auch auf der Ebene der Quelltexte gewährleistet werden, was zudem auch den Grad der Wiederverwendbarkeit erhöht.

Um die Anforderung N3 *Wiederverwendbarkeit* zu erfüllen, wurden alle Komponenten modular implementiert, um lose Kopplung zu erreichen. Beispielsweise bilden alle Klassen, die für die Repräsentation eines OAF Alignment Modells oder für die Ein-/Ausgabe von OAF Dateien verwendet werden, ein Modul. Dieses weist keine Abhängigkeiten zu anderen Komponenten auf und kann daher eigenständig in andere Software integriert werden.

4.2.1 Entity Matching Frameworks

Zur Identifikation von korrespondierenden Entitäten in externen Datensätzen sollte EM eingesetzt werden. Da, wie zuvor in Abschnitt 4.1.3 erläutert wurde, über EM Frameworks abstrahiert werden sollte, musste mindestens ein EM Framework zur Integration in die Erweiterung ausgewählt werden. Da mehrere EM verfügbar waren, wurde eine Bewertung angestellt um festzustellen, welche Frameworks sich eignen. Ein guter Vergleich 11 verschiedener Entity Matching Frameworks ist in [KR09] zu finden. Nach dem Vorbild dieser Arbeit wurde die Einteilung der Frameworks durchgeführt, welche die Grundlage zu deren Bewertung darstellt.

Umfassende quantitative Evaluationsergebnisse verschiedener EM Tools können [FNNS13, GDE⁺13] entnommen werden.

Eine Liste aller 33 Frameworks, die zur Bewertung herangezogen wurden ist

²Eine Beschreibung des OAF, die Spezifikation, sowie eine Java Implementierung sind unter <http://alignapi.gforge.inria.fr/format.html> zu finden. OAF wurde inzwischen mit der Einführung der *Expressive and Declarative Ontology Alignment Language* (EDOAL) erweitert.

dem Anhang E beigelegt. Die Kriterien nach denen bewertet wurde, wie weit existierende Frameworks sich für dieses Projekt eignen, sind in der Reihenfolge ihrer Priorität:

Verfügbarkeit Die EM Software sollte sich lizenzrechtlich in die MediaWiki Erweiterung integrieren lassen. Kann die EM Software nicht zusammen mit der MediaWiki Erweiterung veröffentlicht werden, so muss diese zumindest für Benutzer verfügbar sein.

Funktionsumfang Hinsichtlich des Funktionsumfang wurde untersucht, ob das entsprechende EM Framework Algorithmen zum maschinellen Lernen der Verknüpfungsvorschrift enthält, da dies für die Benutzbarkeit von Bedeutung ist. Der Funktionsumfang sollte die Realisierung der Komponenten *EM Frameworks* und *Referenzverknüpfungen* ermöglichen, deren Einführung in Abschnitt 3.3.3 begründet wurde.

Neben der Möglichkeit Verknüpfungsvorschriften zu lernen, sollte das Framework außerdem offene Standards verwenden, um die Kompatibilität von Ein- und Ausgabedaten zu gewährleisten.

Integrierbarkeit Technisch sollte die EM Software mit möglichst geringem Aufwand in die MediaWiki Erweiterung integriert werden können. Dazu gehört auch, dass möglichst wenig Abhängigkeiten zu weiterer Software existieren.

Status der Entwicklung Es wurde anhand der Veröffentlichungsdaten und -zyklen bewertet, ob ein Projekt noch aktiv entwickelt oder gewartet wird und sofern das möglich war, wie der Stand der Entwicklung ist, durch Angaben der Autoren oder falls vorhanden ein Ticketsystem zur Verwaltung von Implementierungsaufgaben und Bugs.

Die Frameworks, die durch den Auswahlprozess ausgewählt wurden, waren *LIMES* (Link Discovery Framework for metric spaces) [NA11] und *Silk* [VBGK09]. Sowohl LIMES wie auch Silk implementieren Algorithmen zum maschinellen Lernen von Verknüpfungsvorschriften und können Datensätze über SPARQL Endpoints abfragen. Die Möglichkeit RDF Dateien zu verarbeiten bietet jedoch nur Silk an. Wenngleich die Architektur eine Abstraktion über EM Frameworks vorsah und diese beiden Frameworks implementiert werden sollten, konnten bis zum Ende dieser Arbeit aus zeitlichen Gründen nicht beide Frameworks als EM Komponente in die Erweiterung integriert werden. Implementiert im Rahmen dieser Arbeit wurde die Integration von LIMES als EM Framework, das daher im Folgenden kurz vorgestellt wird.

Das EM Framework LIMES nutzt die Dreiecksungleichung für metrische Räume, um die Anzahl notwendiger Vergleichsoperationen zu reduzieren, also eine Form von Blocking anzuwenden, um den Matching Prozess zu beschleunigen. Es enthält die Algorithmen *RAVEN* (RApid active liNking)

[NLAH11] und *EAGLE* [NL12] zum Lernen von Verknüpfungsvorschriften. Beide Algorithmen arbeiten mit den Attributen von Entitäten, wobei zunächst korrespondierende Attribute identifiziert werden, die dann später zum Lernen der Verknüpfungsvorschrift verwendet werden. Im Unterschied zu *EAGLE* lernt *RAVEN* geeignete Distanzmaße für die Attribute nicht, sondern zieht immer nur festgelegte, vordefinierte Distanzmaße heran. Aus diesem Grund wurde *EAGLE* bei der Implementierung der MediaWiki Erweiterung verwendet.

Alle Komponenten der Erweiterung wurden in PHP implementiert, da dies die Programmiersprache von MediaWiki und Semantic MediaWiki war und sich daher gut für eine einfache Integration geeignet hat. Zur Ansteuerung von LINES wurde eine eigens für diese Arbeit und freundlicherweise von Mitarbeitern des Instituts für Angewandte Informatik der Universität Leipzig modifizierte Version des Quelltextes von LINES 0.6.RC3 bereitgestellt. Diese Quelltexte von LINES in Java ermöglichten die Implementierung einer Schnittstelle zu PHP, die in Scala verfasst wurde. Die Schnittstelle erweiterte das CLI von LINES um Steuerungsmöglichkeiten und die Anwendung von *EAGLE* zum unüberwachten Lernen der Verknüpfungsvorschrift.

4.2.2 Datenquellen

Die EM Frameworks, die nach der im vorherigen Abschnitt 4.2.1 dargelegten Entscheidungsfindung eingesetzt werden sollten, erforderten den Zugriff auf die Datensätze über einen SPARQL Endpoint. Dies stellte einen Umstand dar, da SMW standardmäßig keinen SPARQL Endpoint anbietet und wie auch MediaWiki eine SQL Datenbank, anstelle eines Triplestores zum Speichern von Daten verwendete. Weiterhin sollten nach Anforderung F2 *Datensatzabstraktion* auch RDF Dateien verarbeitet werden können. Zwei alternative Lösungswege für diese Probleme sollen im Folgenden diskutiert und anschließend der gewählte Lösungswege beschrieben werden.

Alternative 1

Um diese Probleme zu lösen, bestand die Möglichkeit die Funktion von SMW zu nutzen, neben dem SQL Datenbank Back-End zusätzlich einen Triplestore zu verwenden. RDF Dateien könnten dann in den Triplestore importiert werden. Allerdings ist die ausschließliche Verwendung eines Triplestores durch SMW nicht möglich, die Daten können nur gespiegelt werden. Die gleichzeitige Verwendung zweier Datenbank Back-Ends bringt zwar einige Nachteile durch die Redundanz mit sich, birgt aber auch Vorteile wie z.B. neues Wissen durch Reasoning inferieren zu können.

Alternative 2

Eine weitere Möglichkeit die Probleme zu lösen, stellte die Verwendung von SMW Erweiterungen dar, die SMW Daten über einen SPARQL Endpoint veröffentlichen³. Die Voraussetzung weiterer Software stellt eine zusätzliche Abhängigkeit dar, könnte aber im Vergleich zur zuvor genannten Möglichkeit die Redundanz und die damit verbundenen Probleme vermeiden. Dieser Ansatz hat allerdings zum Nachteil, dass zusätzlicher Aufwand für die Verarbeitung von RDF Dateien notwendig wäre. Darüber hinaus befanden sich die verfügbaren Erweiterungen im Zeitraum dieser Arbeit noch in einem frühen Entwicklungsstadium, was aufgrund anzunehmender folgender Änderungen zu weiterem Aufwand führen könnte.

Implementierte Lösung

Zur SMW Software gehörten auch einige Skripte, die hilfreich bei Wartungsarbeiten sind. Zu diesen Skripten zählte auch ein Skript zum Export von SMW Daten in RDF/XML Dateien. Die implementierte Lösung der Probleme, die zu Beginn beschrieben wurden, verwendet dieses Skript, indem die für die Anreicherung ausgewählten Entitäten zuerst durch das Skript exportiert werden und anschließend in einen existierenden Triplestore mit SPARQL Endpoint geladen werden.

Ein Nachteil bei dieser Lösung ist ein zusätzlicher Verarbeitungsschritt durch den Export der Entitäten in eine RDF/XML Datei. Im Zeitraum der Verarbeitung gilt zudem der gleiche Nachteil wie in der zuerst vorgestellten Lösung durch die entstehende Redundanz. Allerdings werden die Daten nach der Verarbeitung nicht mehr benötigt und können entfernt werden. Ein Vorteil dieser Lösung liegt darin, dass die Flexibilität größer ist, weil kein spezieller Triplestore vorgeschrieben wird und auch, dass beim Import von RDF Dateien alle Datensätze im gleichen Triplestore gespeichert werden können, was die Wartbarkeit erhöht.

Wie zuvor erwähnt werden bei der implementierten Lösung externe Datensätze, die nicht über einen SPARQL Endpoint, sondern als RDF Dateien vorliegen, in den Triplestore geladen und dann über den gleichen SPARQL Endpoint des Triplestores zur Verfügung gestellt, wie die exportierten Entitäten aus der ViFU. Neben RDF/XML unterstützt Virtuoso außerdem unter anderem die RDF Syntaxen N-Triples, Turtle, XML, OWL, TriG und N-Quads.

³Als Beispiel für eine derartige Erweiterung sei hier RDFIO genannt, <http://www.mediawiki.org/wiki/Extension:RDFIO>, abgerufen am 04.07.2014

4.2.3 Datenintegration

In diesem Abschnitt wird die Implementierung von Funktionen beschrieben, die der Integration der Daten in die ViFU dienen, die Entitäten angehören, welche in externen Datensätzen durch den EM Prozess identifiziert werden konnten.

Ergebnismodifikation und Veröffentlichung

Aus der Anforderungsanalyse ging hervor, dass Ergebnisse kontrolliert und gegebenenfalls korrigiert werden müssen. Um die in Abschnitt 3.1.2 festgelegte Anforderung F5 *Ergebniskontrolle* zu erfüllen wurden daher Funktionen in der Komponente zur Verarbeitung und Integration von Matching Ergebnissen, die in Abschnitt 3.3.3 beschrieben wurde, implementiert. Diese Funktionen umfassten das Löschen und Korrigieren von Verknüpfungen zwischen Entitäten, sowie die Möglichkeit Verknüpfungen als Teil eines Matching Ergebnisses als Links mit Typenangabe den Entitäten der ViFU beizufügen.

Parserfunktionen

Neben der vorher genannten Möglichkeit zur Veröffentlichung von Matching Ergebnissen wurde der Parser von MediaWiki für Benutzereingaben um weitere Funktionen erweitert.

Durch den Aufruf der Parserfunktion *#smwenrich* können Informationen zu einem Auftrag ausgegeben werden oder aber Ergebnisse eines Auftrages aufgelistet werden. Auszug 4.1 zeigt den Aufruf der Parserfunktion, um Informationen zu einem Auftrag abzufragen, wobei die Beschriftungen der Daten als Parameter übergeben werden, zusammen mit dem Ausgabeformat.

```
{{#smwenrich: job=1072633873
| print=info
| id=ID
| name=Name
| description=Beschreibung
| selection=Entitaeten
| linkGroup=Referenzlinks
| dataSource=Datenquelle
| progress=Fortschritt
| startDate=Anfangen
| endDate=Abgeschlossen
| format=table
}}
```

Auszug 4.1: Aufruf der Parserfunktion zur Ausgabe von Auftragsinformationen

In Auszug 4.2 ist der Aufruf in der anderen Form zu sehen, bei der Ergebnisse eines Auftrags ausgegeben werden. Der Parameter *print* steuert den Modus dieser der Parserfunktion. Anstelle der ID des Auftrags kann für den *job* Parameter wahlweise auch der Name eines Auftrags eingesetzt werden, um die Bedienung zu erleichtern.

```
{{#smwenrich: job=1072633873
| print=result
| format=table
}}
```

Auszug 4.2: Aufruf der Parserfunktion zur Ausgabe von Ergebnissen

Die Ergebnisse der Aufrufe dieser beiden Parserfunktionen sind in Abbildung 4.1 zu sehen. Dargestellt wird im oberen Bereich das Ergebnis des Aufrufs der Parserfunktion in ihrer ersten Form und im unteren Bereich das Ergebnis eines Auftrags.

Vorlage	Diskussion	Lesen	Bearbeiten	Versionsgeschichte	Suchen
---------	------------	-------	------------	--------------------	--------

Vorlage:MatchingResult		
ID	1072633873	
Name	Some job	
Beschreibung	With a description	
Entitäten	1758145034	
Referenzlinks	811888964	
Datenquelle	452816685	
Fortschritt		
Angefangen	Tue, 10 Jun 2014 15:25:08 +0200	
Abgeschlossen	Fri, 06 Jun 2014 16:24:17 +0200	
	local entity	similarity external entity
	http://testwiki.smw-cora.org/index.php/Spezial:URI-Auf%C3%B6ser/Spieler_Josef	0.98 http://d-nb.info/gnd/17050199X
	http://testwiki.smw-cora.org/index.php/Spezial:URI-Auf%C3%B6ser/Spieler_Josef	0.98 http://d-nb.info/gnd/117483885

Abbildung 4.1: Ergebnisse des Aufrufs der Parserfunktion zum Abrufen von Auftragsinformationen (oberer Teil) und Ergebnissen (unterer Teil). Die Abbildung stammt aus einer frühen Entwicklungsphase, daher werden anstelle von Namen für Entitätsauswahl und andere Auftrags Elemente nur IDs angezeigt und die Datumsangaben für Anfangs- und Endzeitpunkt sind nicht korrekt.

Semantic Web Browser (SWB)

Die MediaWiki Erweiterung SWB 0.4 wurde an zwei Stellen zur Datenintegration eingesetzt. Zum einen bei der Kontrolle und Korrektur von Ergebnissen deren Grafische Benutzerschnittstelle in Abbildung C.3 zu sehen ist, wobei Entitäten visualisiert werden sollten, was durch die Anzeige eines Fensters mit Informationen beim Bewegen des Mauszeigers über die entsprechende Entität realisiert werden sollte. Und zum anderen auch, um die Integration von Daten externer Entitäten nach der Kontrolle von Ergebnissen zu ermöglichen. Hier wurde genutzt, dass SWB beim Vorhandensein von *equivalent URIs*, also Verknüpfungen die aus einem EM Prozess resultiert sind, alle weiteren Tripel zur Darstellung lädt, welche die externe Entität entweder als Subjekt oder als Objekt haben. Dadurch können Daten von externen Entitäten über eine Verknüpfung zusätzlich zu den Daten angezeigt werden, die in der ViFU vorhanden sind, und dadurch integriert werden.

4.2.4 Serverseitige API

Die Komponente *Server API*, deren Einführung in Abschnitt 3.3.3 begründet wurde, stellt eine Schnittstelle zur Steuerung der Funktionen der Erweiterung bereit. Die Schnittstelle besteht aus vier Modulen, die als Erweiterung der MediaWiki API implementiert wurden. Die vier Module kapseln Schnittstellenfunktionen für die Auftragsverwaltung, die Auswahl von Entitäten, für die Referenzverknüpfungen und die Angabe von Datenquellen.

Die Implementierung als MediaWiki API Modul besteht aus der Spezifikation der verwendeten Parameter sowie deren Beschreibungen, einer Beschreibung des Moduls, Beispielen zu dessen Verwendung und der Implementierung der eigentlichen Funktionalität des Moduls. Bei der Implementierung der extern sichtbaren Funktionalität der Schnittstelle eines Moduls wurde das *Interface Segregation Principle* beachtet, sodass die Komplexität reduziert wird, wodurch sich der Grad der Wiederverwendbarkeit und der Wartbarkeit erhöht.

Nach dem *Single Responsibility Principle* erfüllt ein Schnittstellenmodul nur die Aufgabe einer Schnittstelle. Das bedeutet, dass die Funktionalität eines API Moduls auf hoher Ebene implementiert wurde. Ein API Modul implementiert bspw. keinen Zugriff auf die Datenbank oder implementiert nicht die Steuerung von Entity Matching Software. Das Modul ist lediglich für die Bereitstellung einer Schnittstelle verantwortlich, was unter anderem das Auswerten von Parametern beinhaltet. Um die nach außen sichtbare Funktionalität anzubieten, werden andere Module des Back-Ends verwendet. Dadurch wird die Kohäsion erhöht, was zu gesteigerter Wartbarkeit und Wiederverwendbarkeit führt.

4.3 Implementierung des Front-Ends

Das Front-End bilden sowohl alle serverseitigen Klassen, welche die API der Erweiterung implementieren, sowie alle clientseitigen JavaScript Klassen. Da die serverseitige API der Erweiterung bereits im vorherigen Abschnitt 4.2.4 beschrieben wurde, wird in diesem Abschnitt nur auf die clientseitigen JavaScript Klassen eingegangen.

Bei der Implementierung des Clients zur Steuerung der MediaWiki Erweiterung wurden die Paradigmen der objektorientierten und ereignisorientierten Programmierung verfolgt. Die ereignisorientierte Programmierung wurde bei der Implementierung der grafischen Benutzerschnittstelle (GUI) angewendet. Um die Komplexität zu reduzieren und die Wartbarkeit zu erhöhen, erfolgte die Implementierung strukturiert nach dem *Modell View Controller* (MVC) Entwurfsmuster.

4.3.1 Clientseitige API

Zunächst wurden alle serverseitigen API Funktionen auf eine clientseitige, objektorientierte API in JavaScript abgebildet, sodass die serverseitige API durch JavaScript Klassen aufgerufen werden kann. Hierbei wurden Klassen als reine Datenkontainer (*Model*) angelegt, die neben Methoden zum Zugriff auf Attribute nur eine Methode zu deren Serialisierung bereitstellen und Klassen, welche den Zugriff auf die serverseitige API unter Verwendung der Datenkontainer ermöglichen (*Controller*).

Datenkontainer und die clientseitige API der Erweiterung sind modular entworfen und können auch ohne die GUI in anderer Software als JavaScript Schnittstelle für die MediaWiki Erweiterung integriert werden, um die in Abschnitt 3.1.3 festgelegte Anforderung N3 *Wiederverwendbarkeit* zu erfüllen.

Um ein offenes, standardisiertes Format zum Austausch von Daten zwischen Server und Client zu verwenden, wurden Klassen implementiert, welche zur Serialisierung und Deserialisierung des OAF Dateiformates dienen. Dieses Vorgehen erhöht sowohl die Anforderung N2 *Kompatibilität* wie auch N3 *Wiederverwendbarkeit*.

4.3.2 Grafische Benutzerschnittstelle (GUI)

Das nächste Inkrement bei der Implementierung des Front-Ends verwendete dann die zuvor beschriebene clientseitige API der Erweiterung, um die GUI zu implementieren. Beim Entwurf der GUI wurde, wie zuvor erwähnt, das ereignisorientierte Programmierparadigma angewendet. Dies bedeutet, dass der Quelltext der GUI so aufgebaut wurde, dass die Funktionalität der GUI durch die Reaktion auf Ereignisse realisiert wurde, welche durch den Benutzer verursacht werden. Ein solches Ereignis stellt zum Beispiel das Klicken

auf ein GUI Element dar.

Bei der Implementierung der GUI wurden nach dem MVC Entwurfsmuster Methoden zur Darstellung (*View*) von GUI Elementen von denen zur Steuerung der Programmlogik getrennt, wodurch die Komplexität verringert und Änderungen erleichtert werden, was wiederum zur Erfüllung der in Abschnitt 3.1.3 festgelegten Anforderung N3 *Wartbarkeit* beiträgt.

Um die Wiederverwendbarkeit zu erhöhen, wurde die Programmlogik, welche die GUI Elemente erstellt und initialisiert zudem gekapselt, sodass ein Zugriff auf sensible interne Methoden und Attribute nicht möglich oder zumindest stark erschwert wird. Dies soll die Verwendung des Clients für Entwickler vereinfachen und inkorrektur Benutzung vorbeugen.

Abbildung 4.2 zeigt die Auftragsverwaltung der Grafischen Benutzerschnittstelle, welche die Möglichkeit bietet einen Auftrag zur Modifikation auszuwählen oder hinzuzufügen, ihn zu starten oder zu löschen. Aufträge können benannt werden und haben außerdem eine optionale Beschreibung, sowie einen Fortschritt. Anhang C enthält weitere Abbildungen, welche die anderen Sektionen der Grafischen Benutzerschnittstelle zeigen.

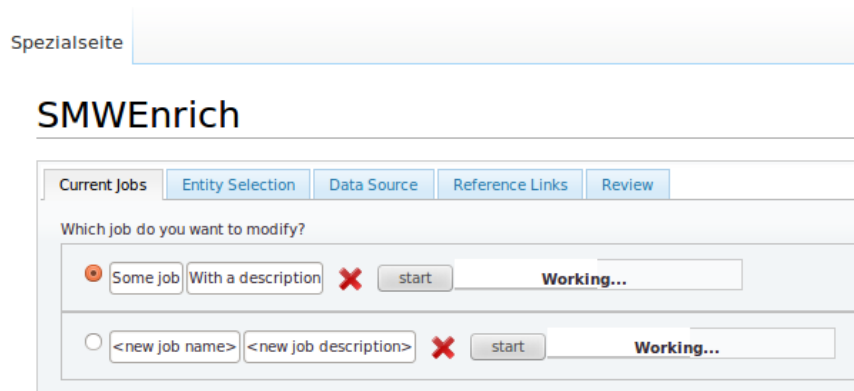


Abbildung 4.2: Auftragsverwaltung der Grafischen Benutzerschnittstelle.

Ereignisse

Ereignisse wie bspw. das Anlegen oder Ändern eines Auftrags oder das Hinzufügen von Entitäten zu einer Auswahl von Entitäten können bei Wiederverwendung in anderer Software trotz der zuvor beschriebenen Kapselung behandelt werden, da die GUI des Clients das Beobachter Entwurfsmuster implementiert. Dadurch wird die Anforderung N3 *Wiederverwendbarkeit* bei gleichzeitiger Kapselung gewährleistet. Die Implementierung der GUI stellt dazu 30 Ereignisarten bereit, durch die Reaktionen auf Benutzerinteraktionen außerhalb der Client Instanz implementiert werden können. Zu den Er-

eignisarten zählen bspw. *initEntityListDone*, *modifiedDataSourceAdded* und *modifiedCurrentJob*. Das Ereignis *initEntityListDone* tritt genau dann ein, wenn die Liste der Entitäten, die bei einem Auftrag zu einer Auswahl an Entitäten gehören, initialisiert worden ist, also alle Entitäten anzeigt, die auf dem Server für diese Auswahl gespeichert sind.

modifiedDataSourceAdded zeigt als Ereignis an, dass eine Datenquelle auf dem Server gespeichert wurde und diese Änderung auch in der GUI des Clients sichtbar gemacht wurde. Das Ereignis *modifiedCurrentJob* ist ein Beispiel für ein komplexes Ereignis, das genau dann eintritt, wenn eine beliebige Änderung an einem existierenden Auftrag gemacht wurde und stellt somit eine Kombination von Ereignissen dar.

Jedes Ereignis hat zudem einen Kontext, der vom Typ des Ereignisses abhängig ist und vom Kontext abhängige Informationen enthalten kann, wie bspw. die ID eines Auftrags, der durch den Benutzer verändert wurde.

4.4 Implementierung spezieller Werkzeuge

Wie in den Grundlagen unter Abschnitt 2.3.4 bereits erwähnt, hängt die Qualität der Ergebnisse eines EM Ansatzes stark vom Anwendungsbereich ab. Aus diesem Grund wurde zu Beginn dieser Arbeit ein spezielles Werkzeug in der Programmiersprache Scala entwickelt, was zunächst verwendet wurde um die Datensätze der ViFU und der DNB zu analysieren.

Neben der Analyse der Datensätze wurde das Werkzeug außerdem entwickelt, um die Datensätze zu partitionieren. Dies sollte zum einen die Entwicklung durch die Verwendung kleinerer Partitionen der Datensätze erleichtern und kann zum anderen in der Vorverarbeitung bei der Durchführung des EM Prozesses eingesetzt werden. Die Partitionierung von Datensätzen ist ein wichtiger Schritt bei parallelen EM Verfahren [KKH⁺10].

Ein weiterer Grund für die Entwicklung dieses Werkzeugs war die Tatsache, dass keine Software zum Partitionieren des verhältnismäßig großen Datensatzes der DNB auf einem Einzelplatzrechner vorhanden war. Existierende Semantic Web Frameworks wie das *Jena*⁴ Framework, *OpenRDF Sesame*⁵ oder die *OWL API*⁶ waren primär so ausgelegt, dass Ontologien wie der Datensatz der DNB, welcher als RDF Datei vorlag, vollständig in den Hauptspeicher geladen werden müssen, bevor Operationen auf der Ontologie ausgeführt werden können. Zwar existierten Ansätze, welche die Verarbeitung größerer Datensätze auf Clustern ermöglichten oder Datenbanken bzw. Triplestores zum Speichern der Ontologien verwendeten, wie bspw. [HKGB09], aber nach ausführlicher Recherche und Tests konnte keine frei verfügbare

⁴<http://jena.apache.org/>, abgerufen am 01.07.2014

⁵<http://www.openrdf.org/>, abgerufen am 01.07.2014

⁶<http://owlapi.sourceforge.net/>, abgerufen am 01.07.2014

Software gefunden werden, die diese Aufgabe ohne weiteren Aufwand auf einem Rechner mit begrenzter Kapazität auszuführen vermochte.

Das implementierte Werkzeug arbeit im Gegensatz zu den oben angeführten Modellen von Jena, der OWL API und Sesame Datenstromorientiert. Dadurch ist zwar nur sequentieller Zugriff möglich, aber der Speicherplatzbedarf während der Verarbeitung ist wesentlich geringer. Um den Nachteil des sequentiellen Zugriffs auszugleichen, wurde die Software so entworfen, dass mehrere Datenströme parallel verarbeitet werden können. Durch die Parallelisierung kann die Geschwindigkeit der Verarbeitung, begrenzt durch Amdahlsches Gesetz, beschleunigt werden. Das Verhältnis zwischen Verarbeitungsgeschwindigkeit und notwendigem Speicherplatzbedarf kann so außerdem grob den verfügbaren Ressourcen angepasst werden.

4.5 Schwierigkeiten

Dieser Abschnitt soll einige Schwierigkeiten hervorheben, die bei der Implementierung aufgetreten sind.

Der Lösungsansatz sah vor, unterschiedliche EM Software zu integrieren. Dieses Vorgehen bringt den Nachteil mit sich, dass die Programmiersprachen und eingesetzten Technologien inhomogen sind und dadurch das Potential für weitere Abhängigkeiten vergrößern. Dieser Umstand musste auch in Bezug auf die Wiederverwendbarkeit berücksichtigt werden.

Als weitere Schwierigkeit kam bei der Implementierung hinzu, dass nicht nur bei der schließlich eingesetzten EM Software entweder keine oder nur unzureichende Dokumentation verfügbar war. Dies lässt sich darauf zurückführen, dass ausschließlich EM Software, die aus dem wissenschaftlichen Bereich stammt, betrachtet wurde und meist entweder als Machbarkeitsbeweis oder zum Vergleich mit anderen Verfahren entwickelt wurde.

Ein weiteres Problem, das sich durch den gewählten Lösungsansatz ergab, war die Einrichtung des SPARQL Zugriffs auf die Daten der ViFU und von externen Datensätzen, die nur in Form von RDF/XML vorliegen, wie es konkret bei der DNB der Fall war. Dadurch, dass existierende EM Software wiederverwendet werden sollte, konnten keine oder nur minimale Anpassungen an den EM Verfahren vorgenommen werden. Da eine der eingesetzten EM Softwares SPARQL Zugriff voraussetzte, gleichzeitig Anforderung F2 *Daten-satzabstraktion* aber auch die Möglichkeit zur Verarbeitung von RDF Dateien verlangte, entstand die Notwendigkeit Datensätze die als RDF Dateien vorliegen über einen SPARQL Endpoint der EM Software bereitzustellen.

Da SMW auf MediaWiki basiert und daher standardmäßig eine SQL Datenbank anstelle eines Triplestores einsetzt, wurde deshalb eine Komponente für die zuvor beschriebene Aufgabe angelegt. Implementiert wurde eine Klasse

zur Verwendung des Triplestores Virtuoso⁷.

Im Zusammenhang mit dem zuvor genannten Problem, verursachte es auch Schwierigkeiten bei der Implementierung, dass SMW / MediaWiki SQL standardmäßig eine SQL Datenbank verwendet. SMW kann um die zusätzliche Verwendung eines Triplestores erweitert werden, was aber in Hinsicht auf die Performanz des Systems unerwünscht sein könnte. Die semantischen Annotation in Artikeln lassen sich von SMW mithilfe eines in SMW enthaltenen Wartungsskriptes in RDF/XML Dateien exportieren, wodurch diese mit der zuvor beschriebenen Komponente als SPARQL Endpoint bereitgestellt werden können.

⁷Virtuoso Open-Source Edition, <http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/>, abgerufen am 04.07.2014

Kapitel 5

Evaluation

In diesem Kapitel wird die Evaluation der Ergebnisse dieser Arbeit dargelegt. Dazu werden zunächst die Ziele der Evaluation festgelegt und anschließend wird geklärt, wie evaluiert wurde. Anschließend werden dann die Resultate der Evaluation vorgestellt und diskutiert. Zuletzt werden dann Probleme angeführt, die nach Abschluß dieser Arbeit noch offen waren und Hinweise gegeben, an welchen Stelle und wie sich diese Arbeit fortsetzen lässt.

5.1 Ziele der Evaluation

Durch die Evaluation dieser Arbeit sollte festgestellt werden, in wie weit die in Abschnitt 1.2 festgelegten Ziele dieser Arbeit erreicht werden konnten. Für den Fall, dass diese nicht erreicht werden konnten, sollten Gründe ermittelt werden, um Schwierigkeiten zu identifizieren.

Ein weiteres Ziel der Evaluation dieser Arbeit war es, Schwierigkeiten zu ermitteln, die bei der Benutzung der MediaWiki Erweiterung auftreten. Dadurch und durch Feedback von Benutzern können Anforderungen weiter verfeinert werden, um die Möglichkeit zu eröffnen, die Entwicklung der Erweiterung fortzusetzen und die Ergebnisse dieser Arbeit weiterzuverwenden.

Es wurde also sowohl eine summative als auch eine formative empirische Evaluation durchgeführt.

5.2 Methodik

Zur summativen Evaluation sollte festgestellt werden, wie weit die Ziele dieser Arbeit erreicht werden konnten und Gründe dafür ermittelt werden, wenn dies nicht möglich war. Dazu wird diskutiert, wie weit die funktionalen und nichtfunktionalen Anforderungen, die in den Abschnitten 3.1.2 und 3.1.3 festgelegt wurden, durch die Realisierung des Lösungsansatzes abgedeckt wurden.

Zur formativen Evaluation wurden Schwierigkeiten, die bei der Benutzung der MediaWiki Erweiterung auftreten, durch einen Usability-Test ermittelt, an dem die ForscherInnen welche die ViFU verwenden, als Probanden teilnahmen. Der Usability-Test und das damit verbundene Interview sollten außerdem der Anforderungsrevision dienen, um Verbesserungsmöglichkeiten zu finden. Für die Durchführung des Usability-Tests wurde die *Co-Discovery* Methode angewendet, bei der im Unterschied zur Methode des *Lauten Denkens* zwei Probanden gemeinsam die Aufgaben bearbeiten und bei der es häufig auch zu mehr Äußerungen kommt als beim lauten Denken [SB11]. Außerdem wurde die *Coaching Method* beim Usability-Test angewendet, vorrangig aus dem pragmatischen Grund, die ForscherInnen gleichzeitig in den Umgang mit der Software einführen zu können und außerdem auch durch die Klärung von Fragen weitere Verbesserungsmöglichkeiten festzustellen. Den Probanden wurde dazu die Aufgabe gestellt, mithilfe der MediaWiki Erweiterung den Standardablauf von Anwendungsfall 1 zu bearbeiten. Das Testsystem wurde den Probanden zur Verfügung gestellt. Die ViFU enthielt im Usability-Test eine kleine Untermenge an Entitäten, die aus Semantic CorA unverändert entnommen wurden. Um die Durchführung des Tests nicht unnötig zu behindern, wurde auch der externe Datensatz auf eine kleine Menge von Entitäten reduziert, welche aus der GND Datei der DNB entnommen wurden. Dadurch wurde die benötigte Rechenzeit verringert, ohne die Bedienung der Erweiterung zu ändern.

Die beiden summativen und formativen Aspekte der Evaluation wurden außerdem durch die Auswertung der Ergebnisse des standardisierten Fragebogens ISONORM 9241/110 ergänzt [J.P97], welcher von den Probanden unmittelbar nach der Durchführung des Usability-Tests ausgefüllt wurde. Die Probanden hatten auch beim Ausfüllen des Fragebogens die Möglichkeit Fragen zu den Funktionen zu stellen und hatten Zugriff auf die Software um sich bestimmte Aspekte noch einmal vor Augen führen zu können.

5.3 Ergebnisse

Dieser Abschnitt stellt die Ergebnisse der Evaluation vor. Zunächst wird bewertet, wie weit die Zielsetzung dieser Arbeit erreicht werden konnte, was nachfolgend mit einer Überprüfung der Abdeckung der festgelegten Anforderungen vertieft wird. Anschließend werden die Ergebnisse aus dem im Rahmen der Evaluation durchgeführten Usability-Test diskutiert und zuletzt die Ergebnisse der Auswertung des Fragebogens ISONORM 9241/110 vorgestellt.

5.3.1 Erfüllung der Zielsetzung

Die in Abschnitt 1.2 festgelegte Zielsetzung konnte weitgehend erreicht werden. Im Rahmen dieser Arbeit wurde ein Lösungsansatz entwickelt, der sowohl aus einer Architektur wie auch aus einer Implementierung unter Anwendung dieser Architektur als Erweiterung für MediaWiki besteht. Dadurch, dass die Erweiterung nicht speziell für die ViFU entwickelt wurde, kann diese in alle ViFUs integriert werden, die auf MediaWiki basieren, womit die Zielsetzung übertroffen wurde.

Nicht erreicht werden konnte der Teilaspekt der Zielsetzung, dass Daten von identifizierten Entitäten semi-automatisch zu denen korrespondierender Entitäten einer ViFU integriert werden können. Als Grund hierfür ist die Annahme zu nennen, dass die ersten beiden Aspekte der Zielsetzung geordnet ablaufen, was in Abschnitt 3.3 erläutert wurde. Aufgrund dieser Annahme wurde der erste Aspekt sowohl theoretisch wie auch praktisch zuerst behandelt, da dieser als Voraussetzung für den zweiten Aspekt erachtet wurde. Zwar wird dieser Aspekt im Lösungsansatz und der Implementierung berücksichtigt, konnte aber aus Zeitmangel nicht vollständig in dem Prototyp der MediaWiki Erweiterung implementiert werden, der im Rahmen dieser Arbeit entwickelt wurde, sodass eine Benutzung oder Evaluation möglich gewesen wäre.

Der dritte Aspekt der Zielsetzung (Benutzbarkeit) konnte zum größten Teil erfüllt werden, wie durch den Usability-Test und die Auswertung des Fragebogens ISONORM 9241/110 gezeigt werden konnte, was in den folgenden Abschnitten beschrieben wird.

5.3.2 Abdeckung von Anforderungen

Die Implementierung umfasst alle Komponenten der Architektur, die in Abschnitt 3.3.3 vorgestellt wurden.

Wie schon im vorherigen Abschnitt erwähnt, konnten die funktionalen Anforderungen F1 bis F5 erfüllt werden. Einzig die funktionale Anforderung F6 *Datenintegration* wurde in Hinsicht auf Anwendungsfall 2 nicht in ausreichendem Maß durch die Implementierung der Lösung abgedeckt.

Bei den nichtfunktionalen Anforderungen konnte im Rahmen der Evaluation gezeigt werden, dass die Anforderung N1 *Benutzbarkeit* größtenteils durch die implementierte Lösung erfüllt wurde. Anforderung N2 *Kompatibilität* wurde durch die Verwendung offener Standards für Ein- und Ausgabeformate und auf Designebene gewährleistet. Die Anforderungen N3 *Wiederverwendbarkeit* und N4 *Wartbarkeit* wurden durch die Anwendung von Designprinzipien abgedeckt, was in Abschnitt 4.1 beschrieben wurde. Die Erfüllung der Anforderung N5 *Skalierbarkeit* wurde nicht im Rahmen der Evaluation dieser Arbeit untersucht, da in Hinsicht auf Skalierbarkeit primär die EM Software untersucht werden müsste und hierzu bereits Arbeiten der Autoren der eingesetzten Software vorliegen, siehe dazu Abschnitt 4.2.1.

5.3.3 Gebrauchstauglichkeit

Um die Gebrauchstauglichkeit zu ermitteln und um die Anforderungen zu verfeinern wurde ein Usability-Test durchgeführt. Der Test sah vor, dass zwei Probanden in Kooperation die Schritte des Standardablaufs von Anwendungsfall 1 bearbeiten sollten. Dabei wurde den Probanden nur dann eine Hilfestellung geboten, wenn diese überhaupt nicht im Stande waren, die Aufgabe zu lösen. Durch Beobachtung konnten Schwierigkeiten identifiziert werden, die im Umgang mit der Software bei auftraten. Fragen der Probanden waren erlaubt.

Unter Hilfestellung konnten die Probanden bis zum Ende des Tests alle Schritte des Standardablaufs von Anwendungsfall 1 absolvieren und damit das Ziel dieses Anwendungsfalls erreichen.

Usability Probleme

Die Diskussion von problematischen Aspekten der Software-Ergonomie während dem Usability-Test bot die Möglichkeit Verbesserungsvorschläge und Kritik zu sammeln. Im folgenden sollen nun die Usability-Probleme zusammenfassend angeführt werden, die durch den Usability-Test und die damit verbundenen Diskussionen ermittelt werden konnten, sowie etwaige Verbesserungsvorschläge angeboten werden.

Unintuitives Hinzufügen von Elementen Das Erstellen eines neuen Auftrags, sowie das Hinzufügen von Entitäten zu einer Auswahl und das Anlegen neuer Entitätslisten verursachte Schwierigkeiten bei den Probanden. Bemängelt wurde, dass die GUI Elemente, wie bspw. Textfelder, die zum Eingeben von Daten für neue Elemente dienten, nicht getrennt von den übrigen GUI Elementen dargestellt wurden und kein expliziter "Hinzufügen" Button vorhanden war. Das Hinzufügen von Elementen wurde daher als unintuitiv empfunden und war den Probanden teilweise nicht ohne Erläuterung möglich. Nach der Erläuterung jedoch, stellte dies kein Problem mehr dar.

Davon ausgenommen ist das Hinzufügen von weiteren Datenquellen, wobei die Textfelder zur Eingabe des Namens und der URL zusammen mit einem "Hinzufügen" Button in einem gesonderten Bereich angeordnet waren.

Fehlende Erklärungen Viele Schwierigkeiten beim Lösen der Aufgaben entstanden dadurch, dass den Probanden nicht klar war, welchen Zweck bestimmte GUI Elemente hatten. Beispielsweise war es zwar möglich, Auswahllisten von Entitäten zu verwalten und diesen weitere Entitäten hinzuzufügen oder diese zu entfernen, führte aber bei den Probanden zu Verwirrungen, weil die Liste der verschiedenen Entitätsauswahlen nicht klar beschriftet war und auch sonst keine Erklärung durch die Software zum Sinn und Zweck dieses GUI Elements geliefert wurde.

Dieses Problem kann relativ einfach durch Erläuterungen und Beschriftungen gelöst werden, die bei den entsprechenden GUI Elementen angezeigt werden. So kam beispielsweise bei der Angabe von Referenzverknüpfungen zunächst die Frage auf, was Referenzverknüpfungen überhaupt sind. Eine Hilfestellung im Programm wie "Die Angabe von Referenzverknüpfungen hilft, Entitäten zu identifizieren" hätte den Probanden den Zweck dieser Funktion erläutern können.

Keine Mehrfachoperationen Während dem Usability-Test wurde mehrfach kritisiert, dass es keine Mehrfachauswahlmöglichkeiten gibt. So fehlt bspw. die Möglichkeit mehrere Entitätslisten zu einer gemeinsamen Liste zu vereinigen, da die Software nur eine exklusive Auswahl zuließ. Die Möglichkeit Vereinigungen von Entitätsauswahlen und Referenzverknüpfungen zu bilden wurde von den Probanden als sehr sinnvoll eingeschätzt.

Unübersichtliche Darstellung Das Layout sorgte durch die teilweise unübersichtliche Darstellung, wenn GUI Elemente oder Funktionale Gruppen von Elementen nicht ausreichend voneinander separiert wurden bei den Probanden zu Schwierigkeiten beim Lösen der Aufgaben. Elemente, die für die Bearbeitung des aktuellen Teilschrittes unwichtig waren wurden mit wichtigen Elementen in Verbindung gebracht, da diese sich in unmittelbarer Nähe zueinander befanden.

Das Layout der Erweiterung kann übersichtlicher gestaltet werden, indem funktional zusammenhängende Elemente gruppiert werden und diese Gruppen voneinander separiert werden.

Der überwiegende Anteil der Schwierigkeiten, die beim Lösen der Aufgaben durch die Probanden auftraten waren auf Unklarheiten der Darstellung von GUI Elementen zurückzuführen. Fehlende oder unverständliche Erklärungen oder Beschriftungen erschwerten den Probanden die Benutzung, was

mehrfach durch diese kritisiert wurde. Neben fehlenden Erklärungen oder Beschriftungen führten auch uneinheitliche Bedienelemente zu Verwirrungen. So erwarteten die Probanden zum Hinzufügen von Objekten Textfelder und ÖK"Buttons, anstelle von editierbaren Textfeldern und automatischer Erkennung neuer Eingaben. Andererseits waren die Probanden im Stande die Aufgaben bei Wiederholung ohne Probleme zu lösen, da ihnen dann die Funktionen und die Mechanismen der grafischen Benutzeroberfläche bekannt waren.

5.3.4 ISONORM 9241/110

Um die Evaluation zu ergänzen wurde weiterhin der durch die Probanden ausgefüllte standardisierte Fragebogen ISONORM 9241/110 (Langfassung) [J.P97] ausgewertet, der sich auf die Normreihe *Ergonomie der Mensch-System-Interaktion*, genauer Teil 110 der Norm DIN EN ISO 9241 bezieht. Diese Norm beschreibt sieben Gestaltungsgrundsätze, die der Fragebogen zur Evaluation heranzieht.

Die verwendete Langfassung des Fragebogens besteht aus 35 Fragen die zur Bewertung von Aussagen über eine siebenstufige Skala von sehr negativ bis sehr positiv dienen und eignet sich sowohl zur summativen Evaluation existierender Software wie auch zur formativen Evaluation von Prototypen und wird von Benutzern der zu evaluierenden Software ohne spezielle Vorbereitung ausgefüllt.

Zur Auswertung wurden die Skalenmittelwerte der einzelnen Fragen über alle Probanden gebildet, die dann wiederum durch ihren Mittelwert zu einer Bewertung des jeweiligen Gestaltungsgrundsatzes zusammengefasst wurden. Abbildung 5.1 stellt den Vergleich mit Referenzwerten aus [J.P97] dar.

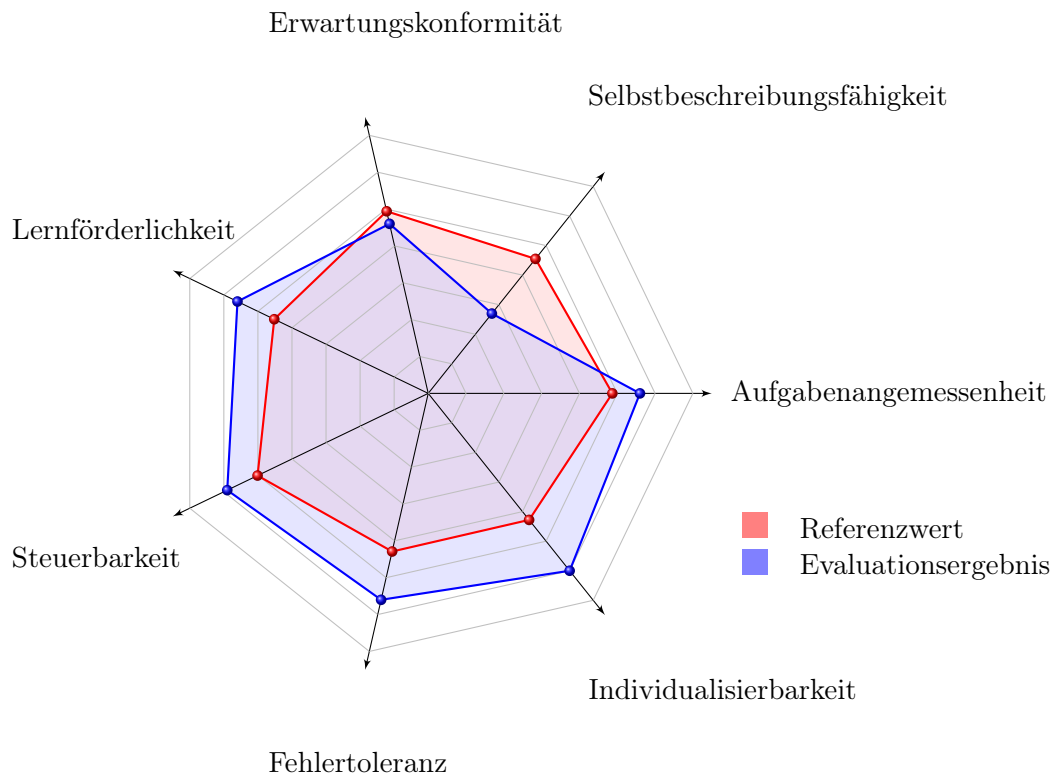


Abbildung 5.1: Ergebnisse der Evaluation mit dem Fragebogen ISONORM 9241/110 (Langfassung): Aufgabenangemessenheit: 5.6, Selbstbeschreibungsfähigkeit: 2.7, Erwartungskonformität: 4.6, Lernförderlichkeit: 5.6, Steuerbarkeit: 5.9, Fehlertoleranz: 5.6, Individualisierbarkeit: 6.0

Der Gesamtmittelwert von 5.14 ist in Bezug auf die ISONORM 9241/110 Skala im positiven Bereich angesiedelt. 5 der 7 Gestaltungsgrundsätze erreichten eine bessere Ausprägung, als die Referenzwerte. Die Erwartungskonformität lag leicht unter dem Referenzwert und die Selbstbeschreibungsfähigkeit wurde deutlich schlechter bewertet im Vergleich zum Referenzwert. Die negative Bewertung der Erwartungskonformität spiegelte sich auch in den Ergebnissen des Usabilit-Tests wieder. Wie im vorherigen Abschnitt 5.3.3 beschrieben ging daraus hervor, dass das Hinzufügen von Elementen unintuitiv war. Die deutlich negative Bewertung der Selbstbeschreibungsfähigkeit der Software kann darauf zurückgeführt werden, dass die Erweiterung keinerlei Erklärungen oder Hilfestellungen anbietet und die Beschriftungen und Beschreibungen von GUI Elementen nicht ausreichend sind. Dies zeigte sich ebenfalls im Usability-Test, wie im vorherigen Abschnitt 5.3.3 beschrieben.

5.4 Offene Probleme

In diesem Abschnitt werden Probleme angeführt, die bei der Evaluation der Implementierung neben den Ergebnissen aus den vorherigen Untersuchungen festgestellt wurden.

5.4.1 Funktionalität der Datenintegration

Die Integration von Ergebnissen eines EM Prozesses wurde zwar implementiert, jedoch bieten sich hier noch weitere Möglichkeiten für Funktionen an, welche die Forscher bei ihrer Arbeit mit der ViFU unterstützen können.

Im Unterschied zu Anwendungsfall 2, der in Abschnitt 3.1.1 beschrieben wurde, weist die Implementierung keine Funktionen auf, Daten bei der Integration in das Vokabular der ViFU abzubilden. Die Back-End Komponente zur Verwaltung von Ergebnissen müsste dafür um entsprechende Funktionen erweitert werden, wobei auch das Front-End um weitere Darstellungs- und Steuerungs-Möglichkeiten ergänzt werden müssten.

Semantic Web Browser

Die Implementierung, die bis zum Ende des Bearbeitungszeitraumes dieser Arbeit erstellt wurde, enthält eine Komponente zur Datenintegration. Diese umfasst auch Funktionen zur Verwendung des *Semantic Web Browser*¹ (SWB) für die Datenintegration, was im Abschnitt 4.2.3 beschrieben wurde.

Bei der Implementierung dieser Funktionen traten besondere Probleme auf, da die Wiederverwendung von SWB 0.4 sich als schwierig gestaltete. SWB ist als Erweiterung für MediaWiki implementiert, wobei die gesamte Funktionalität über eine Spezialseite bereitgestellt wird, wie es auch bei der Erweiterung der Fall ist, die im Rahmen dieser Arbeit entstanden ist. Allerdings ist die aktuelle Version von SWB nicht modular aufgebaut. Sämtliche Funktionen sind in der Klasse implementiert, welche die MediaWiki Spezial Seite bereitstellt. Es gibt keine Trennung von Darstellung und Logik auf dieser Ebene. Weiter sind die einzigen Methoden, die bei Vererbung oder für andere Klassen sichtbar sind, *execute()* und *displayGraph()*, die direkt die HTML Ausgabe zur Darstellung liefern und über keine Parameter zur Steuerung dieser verfügen. Insgesamt führen diese Probleme dazu, dass es nur schwer möglich ist, SWB wiederzuverwenden, da die Ausgabe nicht gesteuert werden kann. Beispielsweise ist es dadurch nicht möglich anstelle von HTML die von SWB aufbereiteten Daten im XML oder JSON Format bereitzustellen.

Diese Probleme konnten jedoch provisorisch durch clientseitige Funktionen

¹ <http://www.mediawiki.org/wiki/Extension:SemanticWebBrowser>, abgerufen am 11.07.2014

umgangen werden, die einen Aufruf der SWB Spezialseite simulieren und das Ergebnis parsen, um dann die gewünschten Daten darzustellen. Um dies zu realisieren müssen jedoch besondere Vorsichtsmaßnahmen getroffen werden, um Benutzer vor Cross-Site-Scripting Attacken zu schützen.

Dieser zusätzliche Aufwand ließe sich vermeiden, wenn die Funktionen zum Parsen und anschließender Ausgabe der SWB Spezialseite serverseitig implementiert würde.

Bis zum Abschluss der Arbeit konnten aus den genannten Gründen die Funktionen zur Datenintegration, die SWB verwenden nicht fertiggestellt werden. Es muss angemerkt werden, dass bisher nur eine Betaversion von SWB verfügbar ist und die beschriebenen Probleme mit zukünftigen Versionen möglicherweise nicht mehr auftreten.

5.4.2 Weitere EM Frameworks

Wenngleich die Architektur eine Abstraktion über EM Frameworks vorsah, konnten bis zum Ende dieser Arbeit nicht alle vorgesehenen Frameworks als EM Software in die Erweiterung integriert werden.

Der Grund für die Abstraktion über EM Frameworks wurde in Abschnitt 4.1.3 erläutert. Die Möglichkeit auf ein anderes EM Framework auszuweichen, sollte das verwendete nicht mehr gewartet werden, wurde jedoch nur als Herausforderung bei der Implementierung betrachtet und nicht als Anforderung festgelegt. Daher konnte das zweite ausgewählte EM Framework *Silk* aus zeitlichen Gründen in Anbetracht anderer Aspekte nicht fertiggestellt werden.

5.4.3 Erweiterung der Konfigurationsmöglichkeiten

Um die Benutzbarkeit zu erhöhen wurden viele technische Details vor dem Benutzer bewusst verborgen. Dazu gehören bspw. die Anzahl der Iterationen beim Lernen von Verknüpfungsvorschriften iterativer maschineller Lernalgorithmen oder die Größe der Population genetischer Lernalgorithmen. An diesen Stellen wurden vordefinierte Werte eingesetzt, die für die verwendeten Datensätze angebracht waren.

Hier wäre möglicherweise eine Evaluation sinnvoll, in wie weit diese vordefinierten Werte Einfluss auf die Ergebnisse haben oder auch Möglichkeiten, diese Werte zu konfigurieren. Zur Evaluation der EM Software könnte das EM Evaluations Framework *FEVER* [KTR09] eingesetzt werden.

5.4.4 Herkunft von Daten

Die in dieser Arbeit vorgestellte Architektur und die zugehörige implementierte Lösung bieten keine Möglichkeiten die Herkunft von Daten zu berücksichtigen. Für die Bearbeitung bestimmter Forschungsfragen könnten Infor-

mationen über die Herkunft bspw. zur Überprüfung von Daten ein sinnvolles Hilfsmittel darstellen. Denkbar wären hierbei auch Anwendungen im Bereich der Rechteverwaltung, um die Verfügbarkeit von Daten einer ViFU unter deren Benutzern zu beschränken.

Kapitel 6

Schluss

Dieses Kapitel schließt diese Arbeit mit einer Zusammenfassung und einem Ausblick auf weiterführende Arbeiten ab. Offene Probleme wurden bereits in der Evaluation festgestellt und sind in Abschnitt 5.4 angeführt.

6.1 Zusammenfassung

Im Rahmen dieser Arbeit sollte eine Lösung entwickelt werden, die Forschern bei ihrer Arbeit mit einer ViFU, am Beispiel von Semantic CorA unterstützt. Der entwickelte Lösungsansatz und dessen Implementierung kann in jede ViFU integriert werden, die auf MediaWiki basiert. Durch den modularen Aufbau können Teile der Architektur leicht ersetzt, erweitert oder in anderer Software wiederverwendet werden. Der Entwicklungsprozess bezog ForscherInnen als Benutzer der ViFU mit ein und wurde so ausgelegt, dass eine langfristige, gemeinschaftliche Entwicklung im Geiste der MediaWiki Gemeinschaft möglich wäre. Die Benutzbarkeit der Softwarelösung wurde als wichtiger Gestaltungsgrundsatz in der Anforderungsanalyse identifiziert und konnte mit positiven Ergebnissen realisiert werden.

6.2 Ausblick

Neben den Gesichtspunkten, die in der Evaluation festgestellt werden konnten, bieten sich weitere Möglichkeiten zur Verbesserung und Fortsetzung dieser Arbeit an. Einige dieser Möglichkeiten werden abschließend in diesem Abschnitt beschrieben.

6.2.1 Kandidaten für Referenzverknüpfungen

Zur Auswahl von Kandidaten für Referenzverknüpfungen existiert Software, welche dem Benutzer geeignete Kandidaten zur Bestätigung präsentiert. Beispiele hierfür sind das EM Framework *Silk* [VBGK09] und *SAIM* [LHS⁺13].

Die Wiederverwendung und Integration derartiger Software könnte den EM Prozess für die Benutzer weiter vereinfachen.

6.2.2 EM Crowdsourcing

Neben der maschinellen Berechnung der Korrespondenzmenge bei EM, wurden auch Ansätze untersucht, die Crowdsourcing zur Lösung dieses Problems verwenden [DDCM12, WOKK14, LNE13, SAN12].

Im Kontext Virtueller Forschungsumgebungen sind diese Ansätze interessant, da Virtuelle Forschungsumgebungen mitunter größere Benutzergruppen haben und sich dadurch ein Potential für deren Anwendung ergibt.

Anhang A

Abkürzungsverzeichnis

API Application Programming Interface
BDI Big Data Integration
CFL Comprehensive, Functional, Layered
CLI Command Line Interface
EM Entity Matching
DFG Deutsche Forschungsgemeinschaft
DIPF Deutsches Institut für Internationale Pädagogische Forschung
DNB Deutsche Nationalbibliothek
EDOAL Expressive and Declarative Ontology Alignment Language
EMO Encapsulated ^{my}Experiment Object
GND Gemeinsame Normdatei
GOV Global-As-View
GUI Graphical User Interface
HTML Hypertext Markup Language
HTTP Hypertext Transfer Protocol
IE Informations Extraktion
IR Information Retrieval
KIT Karlsruher Institut für Technologie
LAV Local-As-View
LOD Linked Open Data
MVC Model View Controller
OAF Ontology Alignment Format
OAI Open Archives Initiative
OAI-ORE Object Reuse and Exchange
OWL Web Ontology Language
PHP Hypertext Preprocessor
RDF Resource Description Framework
RDFS Resource Description Framework Schema
SMW Semantic MediaWiki
SMW-LDE Semantic MediaWiki Linked Data Environment

SWB Semantic Web Browser
TF-IDF Term Frequency–Inverse Document Frequency
ViFU Virtuelle Forschungsumgebung
URI Uniform Resource Identifier
URL Uniform Resource Locator
VRE Virtual Research Environment
W3C World Wide Web Consortium
XML Extensible Markup Language
XSD XML Schema Definition

Anhang B

Quelltexte

```
<?xml version="1.0"?>
<Alignment>
  <xml>yes</xml>
  <level>0</level>
  <type>**</type>
  <Ontology rdf:about="http://some.thing.org/ontology">
    <location>http://some.thing.org/ontology.owl</location>
    <formalism>
      <Formalism align:name="OWL1.0" align:uri="http://www.w3
        .org/2002/07/owl#" />
    </formalism>
  </Ontology>
  <Ontology rdf:about="http://other.thing.org/ontology">
    <location>http://other.thing.org/ontology.owl</location>
    <formalism>
      <Formalism align:name="OWL1.0" align:uri="http://www.w3
        .org/2002/07/owl#" />
    </formalism>
  </Ontology>
  <map>
    <Cell rdf:about="#matchWithinOntology1">
      <entity1 rdf:resource="http://some.thing.org/ontology#
        entity1" />
      <entity2 rdf:resource="http://some.thing.org/ontology#
        entity2" />
      <relation>=</relation>
      <measure rdf:datatype="http://www.w3.org/2001/
        XMLSchema#float">1.0</measure>
    </Cell>
  </map>
</Alignment>
```

Auszug B.1: Beispiel einer Datei im Ontology Alignment Format (OAF) mit einer Verknüpfung von zwei korrespondierenden Entitäten innerhalb einer Ontologie.

Anhang C

Grafische Benutzerschnittstelle

Spezialseite

SMWEnrich

Current Jobs

Entity Selection

Data Source

Reference Links

Review

Which entity list do you want to use?

☒ New selection

Description

✗

☐ New selection

Description

✗

• Spieler, Josef

✗

• Dransfeld, Hedwig

✗

Add entity:

Add category:

ASK query:

Abbildung C.1: Sektion der Grafischen Benutzerschnittstelle zur Verwaltung von Entitätsauswahlen. Das Feld zum Hinzufügen einer Entität verfügt über eine Funktion zur automatischen Vervollständigung. Die halbtransparenten Felder im unteren Bereich zeigen weitere Möglichkeiten Entitäten auszuwählen, die allerdings nicht Teil der Implementierung waren.

SMWEnrich

Current Jobs

Entity Selection

Data Source

Reference Links

Review

Which external data source do you want to use?

☐ dnb

☒ dnb rdf file

Add new data source:

OK

Abbildung C.2: Sektion der Grafischen Benutzerschnittstelle zur Verwaltung von Datenquellen. Datenquellen können hinzugefügt, gelöscht und geändert werden. Sowohl SPARQL Endpoints wie auch RDF Dateien können angegeben werden.

SMWEnrich

Current Jobs

Entity Selection

Data Source

Reference Links

Review

Are these links correct?

http://localhost/smw-cora/index.php/Spezial:URI-Auff%C3%B6ser/Spieler_Josef
← 0.98 →
http://d-nb.info/gnd/117483885

http://localhost/smw-cora/index.php/Spezial:URI-Auff%C3%B6ser/Spieler_Josef
← 0.98 →
http://d-nb.info/gnd/17050199X

Publish

Abbildung C.3: Sektion der Grafischen Benutzerschnittstelle zur Kontrolle und Integration der Ergebnisse eines Auftrags. Zu sehen sind zwei gefundene Verknüpfungen.

SMWEnrich

Current Jobs

Entity Selection

Data Source

Reference Links

Review

Select reference links group

☒ Some Name With a description ✖

☐ New link group Description ✖

☒ Spieler, Josef <http://d-nb.info/gnd/117483885> ✖

Add reference link:

Spieler, Josef

[p://d-nb.info/gnd/117483885](http://d-nb.info/gnd/117483885)

Abbildung C.4: Sektion der Grafischen Benutzerschnittstelle zur Verwaltung von Referenzverknüpfungen. Referenzverknüpfungen können gruppiert werden. Spätere Änderungen an Gruppen und Referenzverknüpfungen sind möglich. Das Feld zur Eingabe der lokalen Entität verfügt über eine Funktion zur automatischen Vervollständigung.

Anhang D

Anforderungsanalyse

Fragebogen

Für jede der Fragen sollte beantwortet werden, ob die entsprechende Lösungsmöglichkeit *Nicht sinnvoll*, *Weniger sinnvoll*, *Eher sinnvoll* oder *Sehr sinnvoll* ist. Grundlage der Fragen, die sich auf verschiedene Aspekte bezogen waren kurz formulierte Beispielszenarien.

Möglichkeiten zum Suchen

Angenommen, es sollen Einträge von Personen in der Deutschen Nationalbibliothek gefunden werden.

Frage 1

Personen sollen in der Virtuellen Forschungsumgebung ausgewählt werden durch...

- ...Kategorien
- ...einen Link auf der Seite einer Person
- ...Eingabe einer Liste von Personen
- ...Kriterien, wie beispielsweise Personen, die in Berlin geboren wurden
- eigener Vorschlag:

Bezugsrahmen einer Suche

Sollen nicht Personen, sondern etwas wonach vorher noch nie gesucht wurde gesucht werden, wie beispielsweise literarische Werke, so erfordert dies aus technischen Gründen genaue Angaben.

Frage 2

Wie häufig soll eine neue Art von Objekt (also bspw. keine Person) gesucht werden?

Verarbeitung von Ergebnissen

Für beispielsweise 100 Personen wurden Geburtsorte gefunden.
Bitte bewerten Sie nun, wie sinnvoll die hier vorgestellten Möglichkeiten zur Verarbeitung der gefundenen Geburtsorte Ihrer Meinung nach sind.
Außerdem haben Sie die Möglichkeit eigene Vorschläge zu machen.

Frage 3

Manuelle Kontrolle und ggf. Korrektur des gefundenen Geburtsortes...

- ...bei jeder einzelnen Person
- ...nur bei bestimmten Personen

Frage 4

Automatische Kontrolle und ggf. Korrektur des gefundenen Geburtsortes...

- ...für ausnahmslos alle Personen
- ...für bestimmte Personen

Frage 5

Semi-automatische Kontrolle und ggf. Korrektur des gefundenen Geburtsortes...

- ...für jede einzelne Person
- ...nur für bestimmte Personen

Frage 6

Sonstige Möglichkeiten

- Gefundene Geburtsorte ohne weitere Bearbeitung eintragen
- Gefundene Geburtsorte nur bei manchen Personen eintragen

Anhang E

Liste betrachteter Entity Matching Frameworks

SAIM
Swoosh
SERF
Silk - Linking Framework
RelDC Engine
Context Based Framework
KnoFuss
FEBRL
COMA++
S-Match++
ASID
Quickmig
Glue
ORCHID
AgreementMakerLight
CIDER-CL
CroMatcher
IAMA
LogMap, LogMapLt
MaasMatch
StringsAuto MapSSS
ODGOMS
RiMOM
ServOMap
SLINT+
SPHeRe (System for Parallel Heterogeneity Resolution)
SYNTHESIS
WeSeE-Match

XMapGen XMapSiG
YAM++
SERIMI
SBUEI
CODI

Literaturverzeichnis

- [ACB03] ADOLPH, STEVE, ALISTAIR COCKBURN und PAUL BRAMBLE: *Use cases effektiv erstellen : [das Fundament für gute Software-Entwicklung, Geschäftsprozesse mit uses cases modellieren, die Regeln für uses cases sicher beherrschen]* / Alistair Cockburn. Übers. aus dem Amerikan. von Rüdiger Dieterle. mitp, 2003.
- [AGK06] ARASU, ARVIND, VENKATESH GANTI und RAGHAV KAUSHIK: *Efficient Exact Set-similarity Joins*. In: *Proceedings of the 32Nd International Conference on Very Large Data Bases*, VLDB '06, Seiten 918–929. VLDB Endowment, 2006.
- [BBEG10] BECKER, CHRISTIAN, CHRISTIAN BIZER, MICHAEL ERDMANN und MARK GREAVES: *Extending SMW+ with a Linked Data Integration Framework*. In: POLLERES, AXEL und HUAJUN CHEN (Herausgeber): *ISWC Posters&Demos*, Band 658 der Reihe *CEUR Workshop Proceedings*. CEUR-WS.org, 2010.
- [BCC03] BAXTER, ROHAN, PETER CHRISTEN und TIM CHURCHES: *A Comparison of Fast Blocking Methods for Record Linkage*. In: *ACM SIGKDD '03 Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, Seiten 25–27, 2003.
- [BCH⁺10] BLANKE, TOBIAS, LEONARDO CANDELA, MARK HEDGES, MIKE PRIDDY und FABIO SIMEONI: *Deploying general-purpose virtual research environments for humanities research*. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 368(1925):3813–3828, 2010.
- [BdMnw12] BÖHM, CHRISTOPH, GERARD DE MELO, FELIX NAUMANN und GERHARD WEIKUM: *LINDA: distributed web-of-data-scale entity matching*. In: CHEN, XUE WEN, GUY LEBANON, HAIXUN WANG und MOHAMMED J. ZAKI (Herausgeber): *CIKM*, Seiten 2104–2108. ACM, 2012.
- [BG04] BRICKLEY, DAN und RAMANATHAN V GUHA: *RDF vocabulary description language 1.0: RDF schema*. 2004.

- [BHBL09] BIZER, CHRISTIAN, TOM HEATH und TIM BERNERS-LEE: *Linked data-the story so far*. International journal on semantic web and information systems, 5(3):1–22, 2009.
- [BKM06] BILENKO, MIKHAIL, BEENA KAMATH und RAYMOND J MOONEY: *Adaptive blocking: Learning to scale up record linkage*. In: *Data Mining, 2006. ICDM'06. Sixth International Conference on*, Seiten 87–96. IEEE, 2006.
- [BL98] BERNERS-LEE, TIM: *Semantic web road map*, 1998.
- [BL00] BERNERS-LEE, TIM: *Semantic Web - XML2000*, 2000. Aus einem Vortrag von Tim Berners-Lee für das W3 Consortium.
- [BL06a] BERNERS-LEE, TIM: *Artificial Intelligence and the Semantic Web: AAAI 2006 Keynote*. W3C Web site, 2006. Abgerufen am 08.07.2014.
- [BL06b] BERNERS-LEE, TIM: *Design issues: Linked data*, 2006.
- [BLHL⁺01] BERNERS-LEE, TIM, JAMES HENDLER, ORA LASSILA et al.: *The semantic web*. Scientific american, 284(5):28–37, 2001.
- [BMS07] BAYARDO, ROBERTO J., YIMING MA und RAMAKRISHNAN SRIKANT: *Scaling Up All Pairs Similarity Search*. In: *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, Seiten 131–140, New York, NY, USA, 2007. ACM.
- [CCP⁺07] CANDELA, LEONARDO, DONATELLA CASTELLI, PASQUALE PAGANO, CONSTANTINO THANOS, YANNIS IOANNIDIS, GEORGIA KOUTRIKA, SEAMUS ROSS, H SCHEK und HEIKO SCHULDT: *Setting the foundation of digital libraries* '. D-Lib Magazine, 13(3/4):1082–9873, 2007.
- [CCP13] CANDELA, LEONARDO, DONATELLA CASTELLI und PASQUALE PAGANO: *Virtual Research Environments: An Overview and a Research Agenda*. Data Science Journal, 12(0):GRDI75–GRDI81, 2013.
- [CKM09] CHEN, ZHAOQI, DMITRI V. KALASHNIKOV und SHARAD MEHROTRA: *Exploiting context analysis for combining multiple entity resolution systems*. In: *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, Seiten 207–218. ACM, 2009.
- [CL96] COWIE, JIM und WENDY LEHNERT: *Information Extraction*. Commun. ACM, 39(1):80–91, Januar 1996.

- [Coh00] COHEN, WILLIAM W: *Data integration using similarity joins and a word-based information representation language*. ACM Transactions on Information Systems (TOIS), 18(3):288–321, 2000.
- [CRF03] COHEN, WILLIAM W., PRADEEP RAVIKUMAR und STEPHEN E. FIENBERG: *A comparison of string distance metrics for name-matching tasks*. Seiten 73–78, 2003.
- [DDCM12] DEMARTINI, GIANLUCA, DJELLEL EDDINE DIFALLAH und PHILIPPE CUDRÉ-MAUROUX: *ZenCrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking*. In: *Proceedings of the 21st international conference on World Wide Web*, Seiten 469–478. ACM, 2012.
- [DRGS09] DE ROURE, DAVID, CAROLE GOBLE und ROBERT STEVENS: *The design and realisation of the myExperiment Virtual Research Environment for social sharing of workflows*. Future Generation Computer Systems, 25(5):561–567, 2009.
- [DS13] DONG, XIN LUNA und DIVESH SRIVASTAVA: *Big data integration*. In: *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, Seiten 1245–1248. IEEE, 2013.
- [EGHW08] EBERSBACH, ANJA, MARKUS GLASER, RICHARD HEIGL und ALEXANDER WARTA: *Wiki*. Nummer 2. Springer Berlin Heidelberg, 2008.
- [ES13] EUZENAT, JÉRÔME und PAVEL SHVAIKO: *Ontology Matching*. Springer, Berlin, 2. Auflage, 2013.
- [FNNS13] FERRARA, ALFIO, ANDRIY NIKOLOV, JAN NOESSNER und FRANÇOIS SCHARFFE: *Evaluation of instance matching tools: The experience of OAEI*. J. Web Sem., 21:49–60, 2013.
- [Fra05] FRASER, MICHAEL: *Virtual research environments: overview and activity*. Ariadne, Issue 44:31–40, July 2005.
- [FS69] FELLEGI, IVAN P und ALAN B SUNTER: *A Theory for Record Linkage*. Journal of the American Statistical Association, 64(328):1183–1210, 1969.
- [FW04] FALLSIDE, DAVID C und PRISCILLA WALMSLEY: *XML schema part 0: primer second edition*. W3C recommendation, Seite 16, 2004.

- [GBVdM07] GERBER, AURONA J, ANDRIES BARNARD und ALTA J VAN DER MERWE: *Towards a semantic web layered architecture*. Proceedings of IASTED International Conference on Software Engineering, SE 2007:353–362, 2007.
- [GDE⁺13] GRAU, BERNARDO CUENCA, ZLATAN DRAGISIC, KAI ECKERT, JÉRÔME EUZENAT, ALFIO FERRARA, ROGER GRANADA, VALENTINA IVANOVA, ERNESTO JIMÉNEZ-RUIZ, ANDREAS OSKAR KEMPF, PATRICK LAMBRIX et al.: *Results of the Ontology Alignment Evaluation Initiative 2013*. In: *Proc. 8th ISWC workshop on ontology matching (OM)*, Seiten 61–100, 2013.
- [GJM09] GLASER, HUGH, AFRAZ JAFFRI und IAN MILLARD: *Managing co-reference on the semantic web*. 2009.
- [GM12] GETOOR, LISE und ASHWIN MACHANAVAJJHALA: *Entity Resolution: Theory, Practice & Open Challenges*. PVLDB, 5(12):2018–2019, 2012.
- [GvdMB08] GERBER, AURONA, ALTA VAN DER MERWE und ANDRIES BARNARD: *A Functional Semantic Web Architecture*. In: BECHHOFFER, SEAN, MANFRED HAUSWIRTH, JÜRGEN HOFFMANN und MANOLIS KOUBARAKIS (Herausgeber): *The Semantic Web: Research and Applications*, Band 5021 der Reihe *Lecture Notes in Computer Science*, Seiten 273–287. Springer Berlin Heidelberg, 2008.
- [Hal01] HALEVY, ALON Y: *Answering queries using views: A survey*. The VLDB Journal, 10(4):270–294, 2001.
- [HB11] HEATH, TOM und CHRISTIAN BIZER: *Linked data: Evolving the web into a global data space*. Synthesis lectures on the semantic web: theory and technology, 1(1):1–136, 2011.
- [HKGB09] HENSS, JÖRG, JOACHIM KLEB, STEPHAN GRIMM und JÜRGEN BOCK: *A Database Backend for OWL*. In: *OWLED*, Band 529, 2009.
- [HNF⁺14] HAWRYLYCZ, MICHAEL, LYDIA NG, DAVID FENG, SUSAN SUNKIN, AARON SZAFAER und CHINH DANG: *The Allen Brain Atlas*. In: KASABOV, NIKOLA (Herausgeber): *Springer Handbook of Bio-/Neuroinformatics*, Seiten 1111–1126. Springer Berlin Heidelberg, 2014.
- [HPPSH05] HORROCKS, IAN, BIJAN PARSIA, PETER PATEL-SCHNEIDER und JAMES HENDLER: *Semantic web architecture: Stack or two*

- towers? In: *Principles and practice of semantic web reasoning*, Nummer 3703, Seiten 37–41. Springer, 2005.
- [IB12] ISELE, ROBERT und CHRISTIAN BIZER: *Learning expressive linkage rules using genetic programming*. Proceedings of the VLDB Endowment, 5(11):1638–1649, 2012.
- [IB13] ISELE, ROBERT und CHRISTIAN BIZER: *Active learning of expressive linkage rules using genetic programming*. Web Semantics: Science, Services and Agents on the World Wide Web, 23:2–15, 2013.
- [IJB10] ISELE, ROBERT, ANJA JENTZSCH und CHRISTIAN BIZER: *Silk Server-Adding missing Links while consuming Linked Data*. In: *COLD*, 2010.
- [IS08] ISAAC, ANTOINE und ED SUMMERS: *Skos simple knowledge organization system primer*. Retrieved June, 20:2008, 2008.
- [Ise13] ISELE, ROBERT: *Learning Expressive Linkage Rules for Entity Matching Using Genetic Programming*. Doktorarbeit, Universität Mannheim, 2013.
- [Jar89] JARO, MATTHEW A: *Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida*. Journal of the American Statistical Association, 84(406):414–420, 1989.
- [J.P97] J.PRÜMPER: *Der Benutzungsfragebogen ISONORM 9241/10: Ergebnisse Zur Reliabilität und Validität*. In: LISKOWSKY, R., B.M. VELICKOWSKY und W. WÜNSCHMANN (Herausgeber): *Software-Ergonomie '97 – Usability Engineering: Integration von Mensch-Computer-Interaktion und Software-Entwicklung*, Seiten 253–261, Stuttgart, März 1997. Teubner.
- [Kal13] KALASHNIKOV, DMITRI V.: *Super-EGO: Fast Multi-dimensional Similarity Join*. The VLDB Journal, 22(4):561–585, August 2013.
- [KKH⁺10] KIRSTEN, TORALF, LARS KOLB, MICHAEL HARTUNG, ANIKA GROSS, HANNA KÖPCKE und ERHARD RAHM: *Data Partitioning for Parallel Entity Matching*. CoRR, abs/1006.5309, 2010.
- [KM06] KALASHNIKOV, DMITRI V. und SHARAD MEHROTRA: *Domain-independent data cleaning via analysis of entity-relationship graph*. ACM Transactions on Database Systems (TODS), 31(2):716–767, 2006.

- [KMC05] KALASHNIKOV, DMITRI V., SHARAD MEHROTRA und ZHAO-QI CHEN: *Exploiting relationships for domain-independent data cleaning*. In: *SIAM International Conference on Data Mining (SIAM SDM)*, Newport Beach, CA, USA, April 21–23 2005.
- [KMW96] KOUZES, RICHARD T, JAMES D MYERS und WILLIAM A WULF: *Collaboratories: Doing science on the Internet*. Computer, 29(8):40–46, 1996.
- [KR09] KÖPCKE, HANNA und ERHARD RAHM: *Frameworks for entity matching: A comparison*. Data Knowl. Eng., 69(2):197–210, 2009.
- [KTR09] KÖPCKE, HANNA, ANDREAS THOR und ERHARD RAHM: *Comparative evaluation of entity resolution approaches with FEVER*. Proceedings of the VLDB Endowment, 2(2):1574–1577, 2009.
- [KTR10] KÖPCKE, HANNA, ANDREAS THOR und ERHARD RAHM: *Evaluation of entity resolution approaches on real-world match problems*. PVLDB, 3(1):484–493, 2010.
- [LC01] LEUF, BO und WARD CUNNINGHAM: *The Wiki way: quick collaboration on the Web*. 2001.
- [LDWF11] LI, GUOLIANG, DONG DENG, JIANNAN WANG und JIANHUA FENG: *Pass-join: A Partition-based Method for Similarity Joins*. Proc. VLDB Endow., 5(3):253–264, November 2011.
- [Len02] LENZERINI, MAURIZIO: *Data Integration: A Theoretical Perspective*. In: *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, Seiten 233–246. ACM, 2002.
- [LHS⁺13] LYKO, KLAUS, KONRAD HÖFFNER, RENÉ SPECK, AXEL-CYRILLE NGONGA NGOMO und JENS LEHMANN: *SAIM–One Step Closer to Zero-Configuration Link Discovery*. In: *The Semantic Web: ESWC 2013 Satellite Events*, Seiten 167–172. Springer, 2013.
- [LNE13] LEHMANN, JENS, TRI QUAN NGUYEN und TIMOFEY ERMILOV: *Can we Create Better Links by Playing Games?* In: *2013 IEEE Seventh International Conference on Semantic Computing, Irvine, CA, USA, September 16-18, 2013*, Seiten 322–329. IEEE, 2013.

- [LTLL09] LI, JUANZI, JIE TANG, YI LI und QIONG LUO: *Rimom: A dynamic multistrategy ontology alignment framework*. Knowledge and Data Engineering, IEEE Transactions on, 21(8):1218–1232, 2009.
- [Mar03] MARTIN, ROBERT CECIL: *Agile Software Development: Principles, Patterns, and Practices*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2003.
- [ME⁺96] MONGE, ALVARO E, CHARLES ELKAN et al.: *The Field Matching Problem: Algorithms and Applications*. In: *KDD*, Seiten 267–270, 1996.
- [ME97] MONGE, ALVARO und CHARLES ELKAN: *An efficient domain-independent algorithm for detecting approximately duplicate database records*. 1997.
- [MMM⁺04] MANOLA, FRANK, ERIC MILLER, BRIAN MCBRIDE et al.: *RDF primer*. W3C recommendation, 10:1–107, 2004.
- [NA11] NGOMO, AXEL-CYRILLE NGONGA und SÖREN AUER: *LI-MES: a time-efficient approach for large-scale link discovery on the web of data*. In: *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three*, Seiten 2312–2317. AAAI Press, 2011.
- [NdM12] NIKOLOV, ANDRIY, MATHIEU D’AQUIN und ENRICO MOTTA: *Unsupervised learning of link discovery configuration*. In: *The Semantic Web: Research and Applications*, Seiten 119–133. Springer, 2012.
- [Ngo12a] NGONGA NGOMO, AXEL-CYRILLE: *Link Discovery with Guaranteed Reduction Ratio in Affine Spaces with Minkowski Measures*. In: CUDRÉ-MAUROUX, PHILIPPE, JEFF HEFLIN, EVREN SIRIN, TANIA TUDORACHE, JÉRÔME EUZENAT, MANFRED HAUSWIRTH, JOSIANEXAVIER PARREIRA, JIM HENDLER, GUUS SCHREIBER, ABRAHAM BERNSTEIN und EVA BLOMQVIST (Herausgeber): *The Semantic Web – ISWC 2012*, Band 7649 der Reihe *Lecture Notes in Computer Science*, Seiten 378–393. Springer Berlin Heidelberg, 2012.
- [Ngo12b] NGONGA NGOMO, AXEL-CYRILLE: *On Link Discovery using a Hybrid Approach*. Journal on Data Semantics, 1(4):203–217, 2012.
- [Ngo13] NGONGA NGOMO, AXEL-CYRILLE: *ORCHID-Reduction-Ratio-Optimal Computation of Geo-spatial Distances for Link*

- Discovery*. In: *The Semantic Web–ISWC 2013*, Seiten 395–410. Springer, 2013.
- [NKAJ59] NEWCOMBE, HOWARD B, JAMES M KENNEDY, SJ AXFORD und ALLISON P JAMES: *Automatic Linkage of Vital Records Computers can be used to extract "follow-up" statistics of families from files of routine records*. *Science*, 130(3381):954–959, 1959.
- [NL12] NGONGA NGOMO, AXEL-CYRILLE und KLAUS LYKO: *Eagle: Efficient active learning of link specifications using genetic programming*. In: *The Semantic Web: Research and Applications*, Seiten 149–163. Springer, 2012.
- [NLAH11] NGONGA NGOMO, AXEL-CYRILLE, JENS LEHMANN, SÖREN AUER und KONRAD HÖFFNER: *Raven–active learning of link specifications*. In: *Proceedings of the Sixth International Workshop on Ontology Matching*, Seiten 25–37, 2011.
- [NLC13] NGONGA NGOMO, AXEL-CYRILLE, KLAUS LYKO und VICTOR CHRISTEN: *COALA – Correlation-Aware Active Learning of Link Specifications*. In: *Proceedings of ESWC*, 2013.
- [NSL14] NGONGA NGOMO, AXEL-CYRILLE, MOHAMED AHMED SHERIF und KLAUS LYKO: *Unsupervised Link Discovery Through Knowledge Base Repair*. In: *Extended Semantic Web Conference (ESWC 2014)*, 2014.
- [NUMDR09] NIKOLOV, ANDRIY, VICTORIA UREN, ENRICO MOTTA und ANNE DE ROECK: *Overcoming schema heterogeneity between linked semantic repositories to improve coreference resolution*. In: *The Semantic Web*, Seiten 332–346. Springer, 2009.
- [RD00] RAHM, ERHARD und HONG HAI DO: *Data cleaning: Problems and current approaches*. *IEEE Data Eng. Bull.*, 23(4):3–13, 2000.
- [SAN12] SIMPERL, ELENA, MARIBEL ACOSTA und BARRY NORTON: *A Semantically Enabled Architecture for Crowdsourced Linked Data Management*. In: *CrowdSearch*, Seiten 9–14. Citeseer, 2012.
- [SB11] SARODNICK, FLORIAN und HENNING BRAU: *Methoden der Usability Evaluation*. Verlag Hans Huber, 2011.
- [SCF04] SNOWDON, DAVID N, ELIZABETH F CHURCHILL und EMMA-NUEL FRÉCON: *Inhabited Information Spaces: Living with your Data*. Springer, 2004.

- [SER12] SCHINDLER, CHRISTOPH, BASIL ELL und MARC RITTBERGER: *Intra-linking the Research Corpus: Using Semantic MediaWiki as a lightweight Virtual Research Environment*. In: MEISTER, JAN CHRISTOPH, KATRIN SCHÖNERT, BASTIAN LOMSCHÉ, WILHELM SCHERNUS, LENA SCHÜCH und MEIKE STEGKEMPER (Herausgeber): *Digital humanities 2012*, Seiten 359–362, Hamburg, 2012. Hamburg University Press.
- [SER13] SCHINDLER, CHRISTOPH, BASIL ELL und MARC RITTBERGER: *Virtual Research Environment SMW-CorA and its Capacities for Interaction in Social Science and Humanities Research - Using the Example of History of Education (received best paper award)*. In: HOBOM, H.-C. (Herausgeber): *Informationswissenschaft zwischen virtueller Infrastruktur und materiellen Lebenswelten. Tagungsband des 13. Internationalen Symposiums der Informationswissenschaft*, Glückstadt, März 2013. vwh. International Symposium of Information Science. ”Best paper award”.
- [Set10] SETTLES, BURR: *Active learning literature survey*. University of Wisconsin, Madison, 52:55–66, 2010.
- [Sod99] SODERLAND, STEPHEN: *Learning Information Extraction Rules for Semi-Structured and Free Text*. Machine Learning, 34(1-3):233–272, 1999.
- [SS11] SMITH, C und K SOTALA: *Knowledge , networks and nations Global scientific collaboration in the 21st century*. Networks, 03/11(RS Policy document 03/11), 2011.
- [SVRV11] SCHINDLER, CHRISTOPH, CORNELIA VEJA, MARC RITTBERGER und DENNY VRANDECIC: *How to teach digital library data to swim into research*. In: GHIDINI, CHIARA, AXEL-CYRILLE NGONGA NGOMO, STEFANIE N. LINDSTAEDT und TASSILO PELLEGRINI (Herausgeber): *I-SEMANTICS*, ACM International Conference Proceeding Series, Seiten 142–149. ACM, 2011.
- [TCC04] TAZZOLI, ROBERTO, PAOLO CASTAGNA und STEFANO EMILIO CAMPANINI: *Towards a semantic wiki wiki web*. In: *3rd International Semantic Web Conference (ISWC2004)*, Hiroshima, Japan, 2004.
- [Ull97] ULLMAN, JEFFREY D: *Information integration using logical views*. In: *Database Theory—ICDT’97*, Seiten 19–40. Springer, 1997.

- [VBGK09] VOLZ, JULIUS, CHRISTIAN BIZER, MARTIN GAEDKE und GEORGI KOBILAROV: *Discovering and Maintaining Links on the Web of Data*. In: BERNSTEIN, ABRAHAM, DAVID R. KARGER, TOM HEATH, LEE FEIGENBAUM, DIANA MAYNARD, ENRICO MOTTA und KRISHNAPRASAD THIRUNARAYAN (Herausgeber): *International Semantic Web Conference*, Band 5823 der Reihe *Lecture Notes in Computer Science*, Seiten 650–665. Springer, 2009.
- [VCL10] VERNICA, RARES, MICHAEL J. CAREY und CHEN LI: *Efficient Parallel Set-similarity Joins Using MapReduce*. In: *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, Seiten 495–506, New York, NY, USA, 2010. ACM.
- [VKV⁺06] VÖLKEL, MAX, MARKUS KRÖTZSCH, DENNY VRANDECIC, HEIKO HALLER und RUDI STUDER: *Semantic wikipedia*. In: *Proceedings of the 15th international conference on World Wide Web*, Seiten 585–594. ACM, 2006.
- [WD07] WILKINS-DIEHR, NANCY: *Special issue: Science gateways—Common community interfaces to grid resources*. *Concurrency and Computation: Practice and Experience*, 19(6):743–749, 2007.
- [WFL10] WANG, JIANNAN, JIANHUA FENG und GUOLIANG LI: *Trie-join: Efficient Trie-based String Similarity Joins with Edit-distance Constraints*. *Proc. VLDB Endow.*, 3(1-2):1219–1230, September 2010.
- [Win99] WINKLER, WILLIAM E: *The state of record linkage and current research problems*. In: *Statistical Research Division, US Census Bureau*. Citeseer, 1999.
- [WLF14] WANG, JIANNAN, GUOLIANG LI und JIANHUA FENG: *Extending String Similarity Join to Tolerant Fuzzy Token Matching*. *ACM Trans. Database Syst.*, 39(1):7:1–7:45, Januar 2014.
- [WMS04] WELTY, CHRIS, DEBORAH L MCGUINNESS und MICHAEL K SMITH: *OWL Web Ontology Language Guide*. W3C Empfehlung, 2004.
- [WOKK14] WANG, JINGJING, SATOSHI OYAMA, MASAHIKO KURIHARA und HISASHI KASHIMA: *Learning an accurate entity resolution model from crowdsourced labels*. In: *Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication*, Seite 103. ACM, 2014.

- [WYW13] WANG, JIAYING, XIAOCHUN YANG und BIN WANG: *Cache-aware Parallel Approximate Matching and Join Algorithms Using BWT*. In: *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, EDBT '13, Seiten 404–412, New York, NY, USA, 2013. ACM.
- [XWL08] XIAO, CHUAN, WEI WANG und XUEMIN LIN: *Ed-Join: An Efficient Algorithm for Similarity Joins with Edit Distance Constraints*. *Proc. VLDB Endow.*, 1(1):933–944, August 2008.
- [XWL⁺11] XIAO, CHUAN, WEI WANG, XUEMIN LIN, JEFFREY XU YU und GUOREN WANG: *Efficient Similarity Joins for Near-duplicate Detection*. *ACM Trans. Database Syst.*, 36(3):15:1–15:41, August 2011.

Index

- EM, 4
- Entity Matching, 6, 10, 11, 16–22, 24, 25, 28–30, 32, 34, 35, 37–40, 44–47, 49, 51, 54, 55, 60, 64, 65, 68
- Semantic CorA, 12, 14, 15, 28, 29, 31, 44, 58
- ^{my}Experiment, 12, 13
- Virtuelle Forschungsumgebung, 1, 2, 6, 12, 17, 26–32, 34, 35, 38, 39, 43, 44, 48, 49, 51, 54, 55, 58, 59, 64, 66–68
- Alignment, 45
- Anforderungen, 27
- Anforderungsanalyse, 27
- Anwendungsfälle, 28
- Blocking, 21, 22
- Collaboratories, 12
- Datenbereinigung, 20
- Datenintegration, 4, 26, 30–32, 38, 40, 49, 51, 59, 64, 65
- Digital Libraries, 12
- Entity Matching, 25
- HTML, 4, 64
- Information Retrieval, 20
- Informations Extraktion, 20
- Inhabited Information Spaces, 12
- Link Specification, 24
- Linkage Rule, 18
- Linked Data, 5, 6, 13, 17, 20, 26
- Matcher, 22, 23
- Media Wiki, 64
- MediaWiki, 8, 9, 11, 15
- nanoHUB, 13
- OWL, 9–11, 17
- Portland Pattern Repository, 8
- RDF, 6, 9, 11, 28, 30
- RDFS, 9, 11
- Referenzverknüpfung, 19, 29, 40, 44–46, 51
- Science Gateways, 12
- Semantic CorA, 15
- Semantic MediaWiki, 3, 8–11, 15
- Semantic Web, 5, 6, 11, 20
- Semantic Web Browser, 51, 64
- Similarity Join, 21, 25
- SKOS, 10, 17
- SMW-LDE, 31, 32
- SPARQL, 6, 28, 30
- Verknüpfungsspezifikation, 24
- Verknüpfungsvorschrift, 18, 21, 24, 32, 40, 46, 47
- Wiki, 1, 8, 38
- WikiWikiWeb, 8, 9
- XML, 4
- XSD, 11